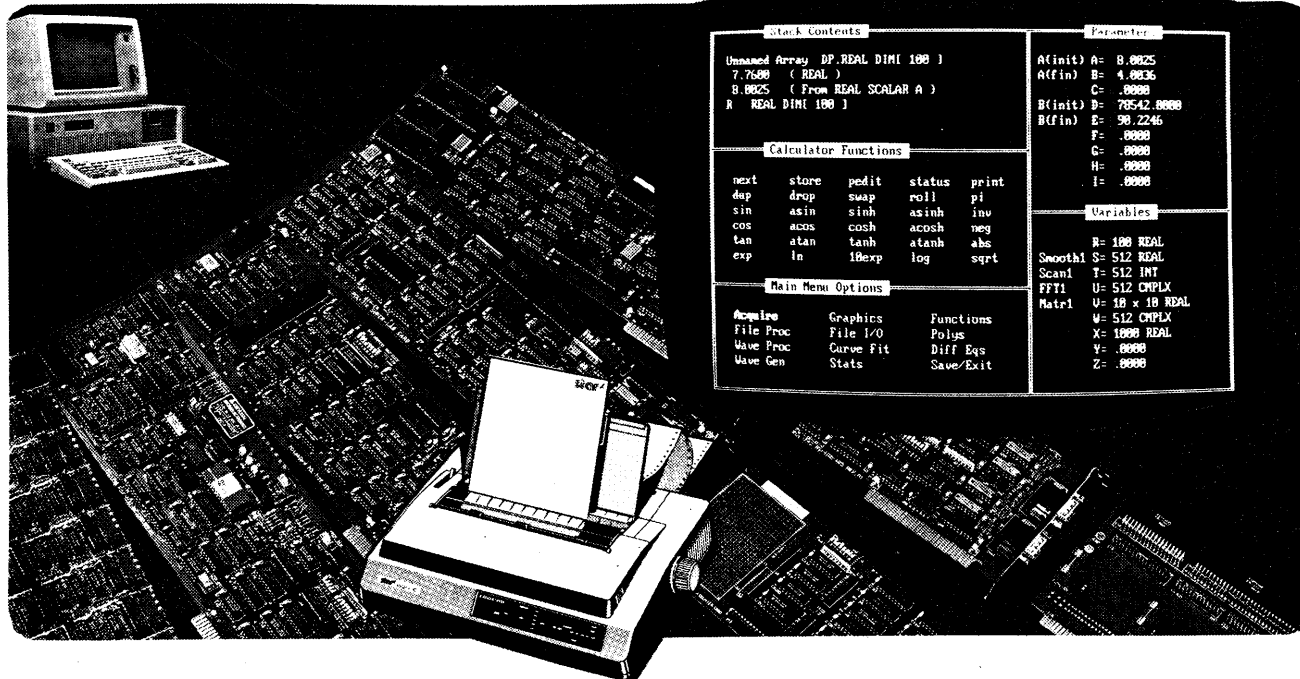


MIKROPROCESOROVÁ A VÝPOČETNÍ TECHNIKA \* HARDWARE & SOFTWARE

# mikroelektronika



Stack Contents		Parameter	
Unused Array DP, REAL DIM 100		A(Init) A= 0.0025	
7.7600 ( REAL )		A(fin) B= 4.0035	
B.0025 ( From REAL SCALAR A )		C= .0000	
A REAL DIM 100		B(Init) D= 70542.0000	
		B(fin) E= 90.2246	
		F= .0000	
		C= .0000	
		H= .0000	
		I= .0000	
Calculator Functions		Variables	
next	store	edit	status
dap	drop	swap	roll
sin	asin	sinh	asinh
cos	acos	cosh	acosh
tan	atan	tanh	atanh
exp	ln	l0exp	log
			sqrt
Main Menu Options		R= 100 REAL	
Acquire	Graphics	Functions	
File Proc	File I/O	Polys	
Wave Proc	Curve Fit	Diff Eqs	
Wave Gen	Stats	Save/Exit	
		Smoother S= 512 REAL	
		Scan1 T= 512 INT	
		FFT1 U= 512 CMPLX	
		Matr1 V= 18 x 18 REAL	
		W= 512 CMPLX	
		X= 1000 REAL	
		Y= .0000	
		Z= .0000	

## ZELENÁ ROČENKA

je stejně jako zelené stránky časopisu Amatérské radio věnována opět tomu z elektronické zájmové činnosti, co souvisí nějak s počítači. Jsou zde převážně příspěvky z minulého ročníku soutěže Mikroprog - Mikrokonkurs, a to zejména ty rozsáhlejší, jejichž publikování v AR na pokračování by bylo nepraktické. Doplnují je drobnější konstrukční i programové výtvořky, tentokrát převážně na téma "propojení počítače s tiskárnou".

Nebude asi mnoho těch, kteří do detailu "okopírují" obzvláště ty rozsáhlé příspěvky. Domníváme se ale, že není jediným účelem zveřejňování navodů, aby je někdo přesně okopíroval. Mnohem větší hodnotu mají jako zdroj inspirace, námětů, dílčích řešení pro úplně jiné úkoly a problémy, nebo třeba jako "učebnice" programování, ap.

Hodně radosti a poučení nad letošní "zelenou ročenkou" vám přeje vaše

redakce AR

### OBSAH:

Klávesnice a displej pro jednočipové mikropočítače	2
Program pro minimalizaci logické funkce	10
Minidata	14
Grafika v PASCALU	16
Sériové prepojenie mikropočítača s tlačiarňou	21
Souřadnicový zapisovač s mikroprocesorem	22
Emulátor terminálů CM7202/CM7209 EMU89	36
Pripojenie tlačiarne K6314	40
Generátor tiskových sestav GTS	42
Hardwarová násobička MH102	44
Pripojenie zapisovača Sharp 1P16	45
Expertík	47
Interfejs tiskárna - Atari	52
Interfejs pro magnetofon k Atari	55
Interfejs Atari 800 - Alfigraf	56
Automatizovaný expoziční systém	58
Minigraf - ZX Spectrum	60
Programování pamětí EPROM na Atari	61
Programátor EPROM 8708	63
Osciloskop zo ZX Spectra	66
Program pro kreslení GEK+	70
BT100 - ZX Spectrum	78



## Úpravy pro použití jednočipových mikropočítačů 8048, 8748

### Zapojení

Jak je zřejmé z obr. 5, je sběrnice použita jako výstupní port pro buzení číslic (DIGIT) v případě, že v aplikacích vystačíme s vnitřními možnostmi mikropočítače. Části portu P2 můžeme použít jako expanderové připojovací vodiče nebo jako CS pro připojení dalších I/O obvodů na sběrnici. Sběrnici a P1 je vhodné připojit na +5 V přes rezistory 10 kΩ.

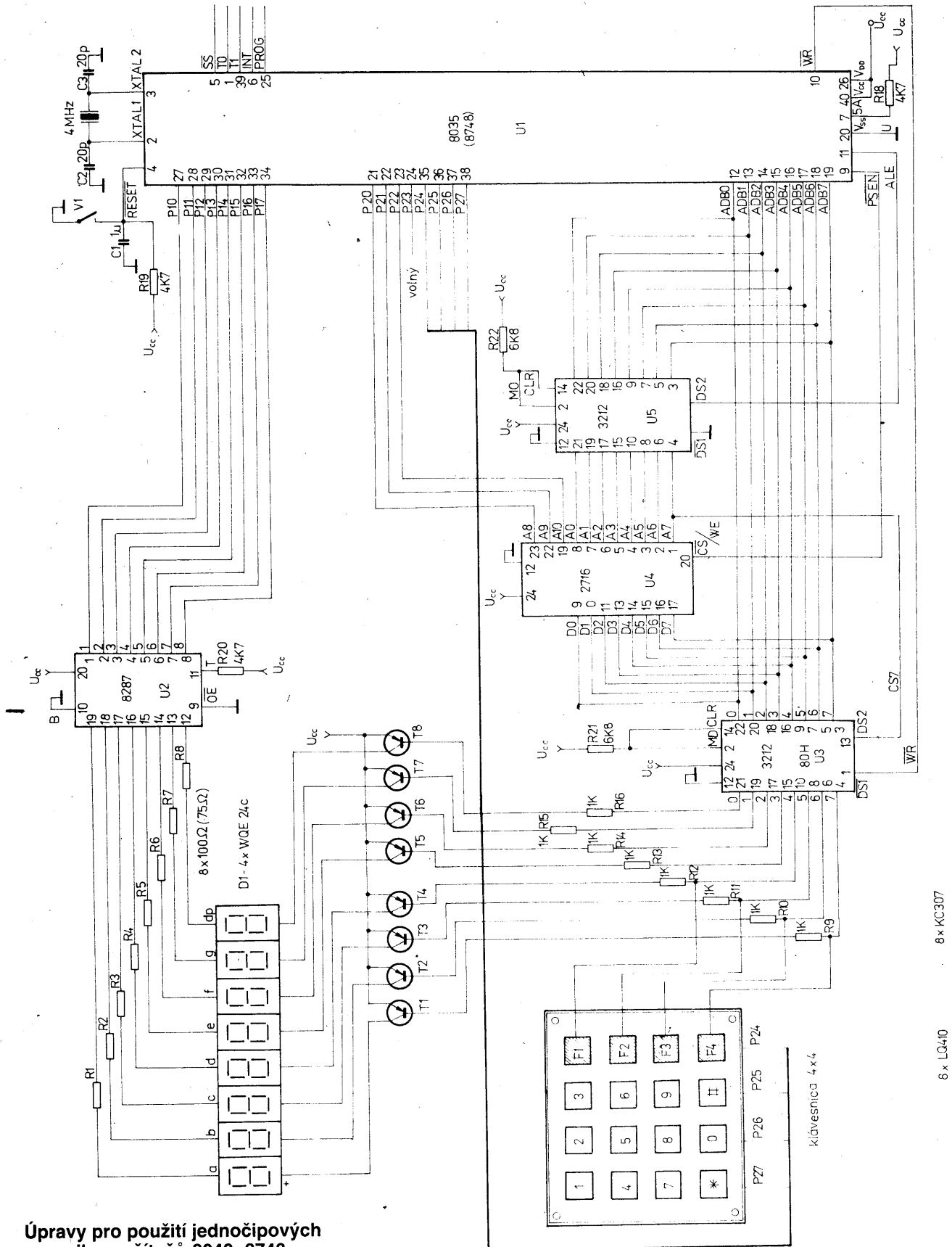
### Programová obsluha

Program se použije stejný, jako pro 8035, jen se vypustí v definiční oblasti registrů banky 0 ACCSA a ROSA (uvolníme si registry R4 a R3 pro použití). Zařadí se instrukce OUTL PDIGIT,A (označena #) v řádku 106 a vypustí se instrukce:

```
SEL RB0
MOV ACCSA,A
MOV A,R0
MOV ROSA,A
MOV R0,#80H
MOV A,ACCSA
```

```
MOV X @RQ,A
MOV A,ROSA
MOV R0,A
MOV A,ACCSA
SEL RB1
(řádky 107 až 117).
```

Obr. 1. Zapojení klávesnice a displeje s 8035 (922-1)



8 x KC307

8 x LQ40





```

1 *****
2 ;
3 ;          INICIALIZACNI PRIRADENI PRO DEFINICI KONFIGURACE
4 ;
5 *****
6 ;
0010 7 PDIGIT EQU      BUS      ;BUDI CISLICE A STROBUJE RADKY KLAVESNICE
0008 8 PSGMNT EQU      P1       ;BUDI SEGMENTY CISLIC DISPLAY V ZAVISLOSTI OD BUZENE CISLICE
0009 9 PINPUT EQU      P2       ;SNIMA STLACENOU KLAVESU BUZENEMO RADKU
10 ;
0000 11 POSLOG EQU      00H     ;POSITIVNI LOGIKA POUZITYCH PRVKU
00FF 12 NEGLOG EQU      0FFH    ;NEGATIVNI LOGIKA POUZITYCH PRVKU
13 ;
00FF 14 CHRPOL EQU      NEGLOG   ;DEF.ZDA VYSTUPNI VODICE JSOU ALTIVNI HI-LO-PDIGIT
0000 15 SEGPOL EQU      POSLOG   ;DEF.ZDA VYSTUPNI VODICE JSOU ALTIVNI HI-LO-PSGMNT
00F0 16 INPMSK EQU      0F0H     ;DEF.BITY POUZITE JAKO VSTUPNI
17 ;
0008 18 CHARNO EQU      8        ;POCET CISLIC DISPLAY (POCET SEMISEGMENTOVEK)
0004 19 NROWS EQU      4         ;POCET RADKU KLAVESNICE(MENSI NEBO = CHARNO)
0004 20 NCOLS EQU      4         ;MENSI DIMENZE KLAVESNICOVE MATICE
21 ;
FFF0 22 TICK EQU      =10H     ;DETERMINUJE INTERRUPT INTERVAL IFRSH RUTINY
0004 23 DEBNCE EQU      4        ;POCET USPESNYCH SNIMANI KLAVESY KU JEJI DETEKCI
0000 24 BLANK EQU      00H      ;KOD MEZERY DISPLAY (BLANK)
25 ;
000F 26 ENCMSK EQU      0FH      ;VYBER KTERE BITY JSOU PLATNE PRI CTENI Z PORTU P2
27 ;
28 *****
29 ;
30 ;          POUZITE REGISTRY BANKY 0
31 ;
32 ;POINTERY PRO ADRESACI PAMETI + USCHOVA ACCU A RO PRI POUZITI 8035
0000 33 PNTR0 EQU      R0        ;
0001 34 PNTR1 EQU      R1        ;
0003 35 ACCSA EQU      R3        ;POUZITO PRO USCHOVU ACU PRI 8035
0004 36 ROSA EQU      R4        ;POUZITO PRO USCHOVU RO PRI 8035
0007 37 NEXTPL EQU      R7       ;UKAZATEL NASLEDNE DISPL.POSICE PRO ZAPIS
38 ;
39 *****
40 ;
41 ;          POUZITE REGISTRE BANKY 1
42 ;
43 ;PNTR0 EQU      R0          ;(UZ BYLO DEFINOVANO)
44 ;PNTR1 EQU      R1          ;
0002 45 ASAVE EQU      R2        ;DRZI HODNOTU ACCU BEHEM REFRESH RUTINY
0004 46 ROTPAT EQU      R4        ;DRZI VSTUPNI VZORKU BEHEM ROTACE PRES CY
0005 47 ROTCNT EQU      R5        ;POCITA BITY PROCHAZEJICI PRES CY
0006 48 LASTKY EQU      R6        ;DRZI POSICI POSLEDNI STLACENE KLAVESY
0007 49 CURDIG EQU      R7        ;DRZI POSICI PRISTHO DISPLAY-ZOBRAZ,ZNAKU
50 ;
51 *****
52 ;
53 ;          DATA RAM ALOKACE
54 ;
0020 55 NREPTS EQU      32       ;DRZI POCET USPESNYCH SEJMUTI DETEKOVANE KLAVESY
0021 56 KEYLOC EQU      33       ;INKREMENTOVANO PRI USPESNEM SEJMUTI KLAVESY
0022 57 KBDUF EQU      34       ;PRENASI POSICI STLACENI KLAVESY Z REFRESH RUTINY
58 ;
0023 59 RDELAY EQU      35       ;NENULOVE PRI FUNKCI REFRESH RUTINY
60 ;
61 ;
0037 62 SEGMAP EQU      (63=CHARNO) ;BASE DISPLAY REGISTRU V INTERNI RAM
63 ;          ; MUZE BYT KDEKOLIV V INTERNI PAMETI RAM
64 ;
65 *****
66 ;
67 ;=====
68 ;
0000 69 ORG      00H          ;
0000 046B 70      JMP      INIT   ;SKOK NA INICIALIZACNI PROCEDURU
71 ;
72 ;
0007 73 ORG      07H          ;
74 ;
75 ;TIINT TIMER INTERRUPT SUBROUTINA
76 ;CALL ADR 007H PRI PRETECENI CASOVACA
77 ;TIMER TADY MUZE BYT PREINICIALIZOVANY PODLE POTIEBY UZIVATELE
78 ;ZDE SE PROVEDE OSVEZENI DISPLAY A SNIMANI KLAVESNICE
79 ;VZDY PO JEDNE POSICI PRI KAZDEM VOLANI
0007 D5 80 TIINT: SEL      R01
0008 AA 81      MOV      ASAVE,A
0009 23F0 82      MOV      A,TICK
0008 62 83      MOV      T,A          ;REINICIALIZACE TIMERU
84 ;
85 *****
86 ;
87 ;          VLASTNI UZIVATELSKA INTERRUPT RUTINA MUZE BYT ZDF UMISTENA
88 ;          PRO LIBOVOLNE POUZITI VE VAZBE NA PRERUSENI CASOVACE
89 ;
90 *****
000C 1410 91      CALL   REFRSH   ;VOLANI DISPLAY/KLAVESNICOVE SUBROUTINY
92 ;TIRET OBNOVA ACCU PRED NAVRATEM Z REFRESH SUBROUTINY
000E FA 93 TIRET: MOV      A,ASAVE
000F 93 94      RETR
95 *****
96 ;
97 ;REFRSH SUBROUTINA MULTIPLEXUJE 7 SEG.DISPLAY A VZDY ZOBRAZI DAI SI POSICI
98 ;NA DISPLAY,JE VHODNE JI VOLAT V MSEC.INTERVALECH
99 *****
100 ;
0010 2300 101 REFRSH: MOV      A,#BLANK XOR SEGPOL
0012 39 102      OUTL     PSGMNT,A      ;ZAPIS BLANK KODU DO PSGMNT
0013 2362 103 REFR1: MOV      A,#CHRSTB
0015 6F 104      ADD      A,CURDIG
0016 A3 105      MOVP     A,SA
106 ;## 106      OUTL     PDIGIT,A      ;BUZENI ZNAKU PRI POUZITI 8748
0017 C5 107      SEL      R01          ;PRO BANK VYBER JEN PRO 8035
0018 AB 108      MOV      ACCSA,A      ;USCHOVA ACCU DO ACCSA JEN PRO 8035
0019 F8 109      MOV      A,R0          ;USCHOVA RO DO ROSA JEN PRO 8035

```

```

001A AC      110      MOV      ROSA,A      ;PRO 8035
001B B880    111      MOV      R0,#080H    ;BIT 7 SET JAKO ADRESA 3212 PRO INSTR,MOVX @R0,A
001D FB      112      MOV      A,ACCSA     ;OBNOVA ACCU PRI 8035
001E 90      113      MOVX     @R0,A      ;ZAPIS DO DIGIT LATCH (3212) JEN PRO 8035
001F FC      114      MOV      A,ROSA     ;OBNOVA R0 JEN PRO 8035
0020 A8      115      MOV      R0,A      ;PRO 8035
0021 FB      116      MOV      A,ACCSA     ;OBNOVA ACCU JEN PRO 8035
0022 D5      117      SEL      R81        ;VYBER FB1 PRO 8035
118 ;
119 ; PRI POUZITI 8748 SE INSTR.OD SEL R80 PO SEL R81 VYPOUSTEJI (JEN PRO 8035)
120 ;=====
121      MOV      A,#SEGMAP    ;DEFINICE BAZE DISPLAY-REGISTRU V RAM
122      ADD      A,CURDIG     ;ADD CURDIG POSINITI
123      MOV      PNTR1,A
124      MOV      A,@PNTR1     ;PLADOVANI VZORKY
125      OUTL     PSGMNT,A     ;BUDI PRISLUSNE SEGMENTY
126 ;
127 ;*****
128 ; DALSI ZNAK JE NYNI ZOBRAZEN NA DISPLAY
129 ; KLAVERNICOVA SNIMACI RUTINA JE ZARADENA DO DISPLAY PROCEDURY
130 ; S VYUZITIM BUZENI PRISLUSNE CISTICE A TIM I PRISL.HADKU KLAUV.
131 ;*****
132 ;
133 SCAN1: MOV     PNTR0,#KEYLOC ;NASTAVUJE POINTER NA UFAZATEL POCTU SNIMANI.
134      IN       A,PINPUT     ;NACITA STLACENI KLAUESY
135 ;
136 ;*****
137 ;
138 ;*****
139 ; ROTACE ACCU PRES CY
140 ;*****
141 SCAN1: MOV     ROTCNT,#NCOLS ;NASTAVUJE ROTACNI SMYCYU PODLE POCTU SLOUPCU
142      NXTLOC: RLC      A
143      MOV      JC      ROTPAT,A      ;UCHOVAVA POSUNITU VZORKU
144      SCANS      ;JEDNOTKOVY BIT INDIKUJE ZE NEBYLA STLACENA KL.
145 ;
146 ;*****
147 ; V TOMTO BODE PROGRAMU BYLO PRAVE ZJISTENO ZE HODNOTA V KEYLOC JE POSICE
148 ; TLACITKA KTERE JE NYNI STLACENE,NASLEDUJICI INSTRUKCE DEKODUJI A URCUJI
149 ; VYZNAM ZMACKNUTHEHO TLACITKA,JAK POUZIVAME MODIFIKACNI VYZNAM FUNKCE
150 ; TLACITKA(SHIFT,CONTROL)NEBO JINY MODALNI VYZNAM TLACITEK MELO IY SE TO
151 ; PROVADET V TOMTO BODE PROGRAMU,PREDTIM NEZ ZACNE DALSI DEKODOVACI LOGIKA
152 ; NAPRIKAD V TETO CASTI MUZE BYT KEYLOC=PROMENNA POROVNAVANA S NAMI
153 ; URCENOU POSICI MODALNIHO TLACITKA A PRI JEHO IDENTIFIKACI SI NASTAVIME
154 ; NEJAKY FLAG S KOCIME NA NAV."SCANS"A NEBO POROVNAVAT PROM,KEYLOC S
155 ; PREDSELYM JEJIM OBSAHEM
156 ;*****
157      CLR      F1          ;ZNACKA ZE NEJMENE JEDNA KLAUESA BYLA DETEKOVANA
158      CPL      F1          ;NEGOVANO V BEZIEH SNIMANI
159 ;
160 ;*****
161 ; ZMACKNUTI TLACITKA BYLO DETEKOVANO V NEKTEREM SLOUPCI JEHO
162 ; JEHO POSICE JE V KEYLOC ZJISTUJE SE JESTLI TO SAME BYLO
163 ; STLACENE V MINULEM CYKLU
164 ;*****
165 ;
166      MOV      A,@PNTR0     ;PNTR0 STALE DRZI #KEYLOC
167      XCH      A,LASTKY
168      XRL      A,LASTKY
169      MOV      PNTR0,#NREPTS ;PRIPRAVA NA KONTROLU
170      ;OPAKOVACI CITAC
171      JZ      SCANS3
172 ;
173 ;*****
174 ; JINE TLACITKO BYLO PRECTENO V TOMTO CYKLE NEZ PFEDTIM
175 ; NASTAVUJE NREPTS NA POZADOVANY PARAMETR PRO NOVI COUNTDOWN
176 ;*****
177 ;
178      MOV      @PNTR0,#DEBNCE
179      JMP      SCANS5
180 ;
181 ;*****
182 ; TO SAME TLACITKO BYLO DETEKOVANO JAKO V PREDSELYM CYKLU
183 ; PODIVEJ SE NA NREPTS JESTLI JE NULOVE,NEDELEJ NIC
184 ; JESTE DEKREMENTUJ NREPTS
185 ; JE LI NULA PRESUN LASTKY DO KBDUF
186 ;*****
187 ;
188 SCAN3: MOV     A,@PNTR0
189      JZ      SCANS      ;JE LI NULA
190      DEC     A          ;INDIKUJE JEDNO NAVIC USPESENE TLAC.ZMACKNUTI
191      MOV     @PNTR0,A
192      JNZ     SCANS      ;JAK VYSLEDEK DEKREMENTACE JE NENULOVOY
193      MOV     A,LASTKY
194      MOV     PNTR0,#KBDUF
195      MOV     @PNTR0,A    ;OZNACUJE NOVE STLACENI TLACITKA
196 ;
197 SCAN5: MOV     PNTR0,#KEYLOC
198      INC     @PNTR0
199      MOV     A,ROTPAT
200      DJNZ    ROTCNT,NXTLOC
201 ;
202 SCAN6: DJNZ    CURDIG,SCAN9
203 ;
204 ;*****
205 ; NASLEDUJICI INSTRUKCE JSOU POUZIVANY KLAVERNICOVOU RUTINOU
206 ; (SCANNING)A JSOU VYKONAVANY POUZE PO REFRESH CELFHO DISPLAY
207 ; TEDY VSECH CISEL DANYCH CHARNO
208 ;*****
209 ;
210      MOV     CURDIG,#CHARNO
211      MOV     @PNTR0,#0
212      JF1     SCANS      ;PNTR0 STALE OBSAHUJE #KEYLOC
213      MOV     LASTKY,#OFFH ;JUMP JAK NEJAK TLACITKO BYLO DETEKOVANO
214      SCAN8: CLR     F1      ;MENI (LASTKY) JEK NEBYLO STLACENO ZADNE TL.
215 ;
216 ;*****
217 ; DALSI PROGRAMOVY SEGMENT JE INTERRUPT-RIZENA CAST "DELAY"
218 ; POUZIVAJIC DEKREMENTACI RAM LOKACE "RDELAY" 1-KFAT ZA DISPL.
219 ; REFRESH DOKUD "RDELAY" NENI NULA

```

```

220 ;*****
221 ;
005B 8923 222          MOV          PNTR1,#RDELAY
005D F1    223          MOV          A,@PNTR1
005E C662 224          JZ          SCAN9
0060 07    225          DEC          A
0061 A1    226          MOV          @PNTR1,A
0062 83    227 SCAN9: RET
228 ;
229 ;*****
230 ;
0062 231 ;CHRSTB JE BAZE VRCHOLU BINARNICH VZOREK POVOLUJICI SVIT 1=SEDMISEGMENTOVKY
232 CHRSTB EQU          ($-1) AND 0FFH
233          DB          (00000001B XOR CHRPOL)
0063 FE    234          DB          (00000010B XOR CHRPOL)
0064 FD    235          DB          (00000100B XOR CHRPOL)
0065 FB    236          DB          (00001000B XOR CHRPOL)
0066 F7    237          DB          (00010000B XOR CHRPOL)
0067 EF    238          DB          (00100000B XOR CHRPOL)
0068 DF    239          DB          (01000000B XOR CHRPOL)
0069 BF    240          DB          (10000000B XOR CHRPOL)
006A 7F    241 ;
242 ;INIT INITIALIZACE PROCESOROVYCH REGISTRU
243 INIT:  SEL          RB1
244          MOV          CURDIG,#CHARNO
006B D5    245          MOV          PNTR0,#KBDDBUF
006C BF08 246          MOV          @PNTR0,#0FFH
006E 8822 247          MOV          PNTR0,#KEYLOC
0070 80FF 248          MOV          @PNTR0,#0
0072 8821 249          MOV          A,#INPMASK
0074 B000 250          OUTL          PINPUT,A          ;NASTAVUJE PINPUT NA LOG.1 PRO STANOVANI DO 0
0076 23F0 251          SEL          R80
0078 3A    252          CALL          CLEAR          ;INICIALIZACE DISPLAY REGISTRU V RAM
0079 C5    253          CLR          F1
007A 14A9 254          MOV          A,#TICK          ;NASTAVENI INTERRUPT INTERVALU
007C A5    255          MOV          T,A
007D 23F0 256          STRT          T
007F 62    257          EN          TCNTI          ;POVOLUJE TIMER INTERRUPT
0080 55
0081 25
258 ;
259 ;
260 ;*****
261 ;
262 ;ECHO KONTROLA ZDA NEJAKE NOVE STLACENI TLAC.BYLO DETEKOVANO
263 ; TRANSFORMUJE KAZDE STLACENI DO DISPILSEGMENTOVE VZORKY
264 ; A ZAPISUJE TO DO PRISLUSNEHO DISPLAY REGISTRU V RAM
265 ;*****
0082 148E 266 ECHO: CALL          KBDIN          ;ZJISTENI NOVEHO STLACENI
0084 B28C 267          JBS          FKEY          ;SKOK JE LI DETEKOVANO TL.V PRAVEM SLOUPCI
268 ; PROTOZE ACCU JE POUZIVANY ENCACC A KENTRY RUTINAMI JEHO OBSAH
269 ; MUSI BYT ZPRACOVANY NERO USCHOVANY PRED VOLANIM ENCACC
0086 14C5 270          CALL          ENCACC          ;TVORI PRISLUSNOCU VZORKU SEGMENT ZOBRAZENI
0088 14E6 271          CALL          RENTRY          ;ZAPISUJE VZORKU DO DISPLAY REGISTRU V RAM
008A 0482 272          JMP          ECHO          ;SMYCKA
273 ;
008C 241F 274 FKEY: JMP          FUNCTN          ;JUMP MIMO STRANKU (OF PAGE CODE) DO DEMONSTR.
275          ;RUTIN RESP.UZIVATELSKY DEFINOVANYCH
RUTIN
276 ;*****
277 ; IMPLEMENTACE NASLEDUJICICH SUBROUTIN SE SPOLECNÍ POUZIVA
278 ; PRO VETSINU KLAVESNICOVE/DISPLAY ORIENTOVANYCH APLIKACI
279 ; MOHOU BYT POUZITE PRESNE JAK JSOU UVEDENE NEBO PRISPUSORENE
280 ; PRO SPECIALNI POUZITI
281 ;*****
282 ;KBDIN KLAVESNICOVA VSTUPNI SUBROUTINA
283 ; MUZE BYT POUZITA KU SPOJENI UZIVATELSKEHO PROGRAMOVEHO POZADI
284 ; S PRERUSEM RIZENOU KLAVESNICOVOU SUBROUTINOU (SCANNER)
285 ; DAVA POUZE PO STLACENI NOVEHO TLACITKA DEKODOVACI OU HODNOTU
286 ; (LEPE NEZ JEHO POSICI V MATICI TLACITEK KLAVESNICE) V ACCU
287 ;
008E B922 288 KBDIN: MOV          PNTR1,#KBDDBUF
0090 2380 289          MOV          A,#80H          ;KBDDBUF BUDE MASKOVAN JAKO CISTY
0092 21    290          XCH          A,@PNTR1          ;NAPLNEFI BUFFROVANOU HODNOTOU
0093 F28E 291          JBS          KBDIN
0095 0399 292          ADD          A,#LEGND5          ;PRICTENI BASE I DEKODOVACI TABULKY TLACITEK
0097 A3    293          MOVP          A,@A          ;OBRZENI PRISLI SNEHO VYZNAMU TLACITKA
0098 83    294          RET
295 ;
296 ;
297 ;LEGND5 JE BAZE TABULKY UKAZUJICI VYZNAM TLACITEK MATICE KLAVESNICE
298 ; PRO KLAVESNICI POUZITOU V TETO UKAZCE APLIKACE
299 ; S USPORADANIM PODLE OBRÁZKU OBR.1
300 ; BITY 6-4 MOHOU BYT POUZITY POUZITY K DEKODOVANI TYPI TLACITEK
301 ; S TIMTO VYZNAMEM:
302 ;
303 ;
304 ;
0099 305 LEGND5 EQU          ($ AND 0FFH)
0099 4F    306          DB          4FH
009A 10    307          DB          10H
009B 4E    308          DB          4EH
009C 28    309          DB          28H          ; PDIGIT4==> 1 2 3 <1>
009D 17    310          DB          17H          ; PDIGIT5==> 4 5 6 <2>
009E 18    311          DB          18H          ; PGIGIT6==> 7 8 9 <3>
009F 19    312          DB          19H          ; PDIGIT7==> * 0 # <4>
00A0 24    313          DB          24H
00A1 14    314          DB          14H
00A2 15    315          DB          15H
00A3 16    316          DB          16H
00A4 22    317          DB          22H          ; 1 1 1 1
00A5 11    318          DB          11H          ; 1 1 1 1
00A6 12    319          DB          12H          ; V V V V
00A7 13    320          DB          13H          ; PINPUT7 PINPUT6 PINPUT5 PINPUT4
00A8 21    321          DB          21H
322 ;
323 ;*****
324 ;CLEAR ZAPISUJE 'BLANK' ZNAK DO DISPLAY REGISTRU V RAM
325 ; VRACI SE S NASTAVENYM NEXTPL NA LEVOU KRAJNI POSICI DISPLAY
326 ;FILL ZAPISUJE SEGMENTOVOU VZORKU V ACCU DO DISPLAY REGISTRU V RAM
00A9 2300 327 CLEAR: MOV          A,#BLANK XOR SEGPOL
00AB 8938 328 FILL: MOV          PNTR1,#SEGMAP+1

```

```

00AD BF08      329      MOV      NEXTPL,#CHARNO
00AF A1        330 CLR1:  MOV      @PNTR1,A          ;UKLADA BLANK KOD
00B0 19        331      INC      PNTR1          ;DALSI POSICE DO LEVA
00B1 EFAF      332      DJNZ    NEXTPL,CLR1
00B3 BF08      333      MOV      NEXTPL,#CHARNO
00B5 83        334      RET
335 ;
336 ;*****
337 ;
338 ;PRINT SUBROUTINA KE KOPIROVANI BINARNICH VZOREK Z "ROM" DO
339 ;      DISPLAY REGISTRU V RAM POCINAJE ADRESOU V PNTR0
340 ;      DOKUD SE NEDDSAHNE ZAKONCOVACI KOD (V TOMTO PRIFADE -00H=)
341 ;      !!! RETEZEC ZNAKU MUSI BYT NA YEZE STRANCE JAKO SUBROUTINA
342 ;      PRINT Z DUVODU VAZBY NA VOLANI "WDISP NEBO RENTFY"
343 ;      PRO EFEKTNI ZAPISY DO DISPLAY REGISTRU V RAM
00B6 F8        344 PRINT:  MOV      A,PNTR0          ;ZADAVANI NASLED.ZNAKOVE POSICE
00B7 A3        345      MOVP    A,@A          ;INACTENI BITOVE VZORKY
00B8 C601      346      JZ      PNTR1          ;DEKODOVANI ZAVERECEHO PRVKU (00H)
00BA 14DB      347      CALL   WDISP          ;VYSTUP NA DALSI ZNAKOVOU POSICI
348 ;##        348      CALL   RENTRY          ;NEBO SKOK NA RENTRY
349 ;
00BC 18        350      INC      PNTR0          ;INDEX UKAZOVATEL
00BD 04B6      351      JMP     PRINT
00BF 83        352 PRNT1:  RET
353 ;
354 ;*****
355 ;
356 ;AH0J OBLAST PAMETI ROM S POZDRAVNOU ZPRAVOU 'AH0J' ( 'TEST2')
357 ;      (UPOZORNENI ZE POUZE "HOJ" SE VYPISE V PRIPADE RENTRY)
00C0          358 AH0J   EQU      $ AND OFFH
00C0 77        359      DB      01110111B XOR SEGPOL
00C1 76        360      DB      01110110B XOR SEGPOL
00C2 3F        361      DB      00111111B XOR SEGPOL
00C3 1E        362      DB      00011110B XOR SEGPOL
00C4 00        363      DB      00
364 ;
365 ;*****
366 ;
367 ;ENCACC DEKODUJE LSBIBLE(SPODNI CAST ACCU)NA HEXA KOD
00C5 530F      368 ENCACC: ANL      A,#ENCMASK
00C7 03CB      369      ADD     A,#DGPATS
00C9 A3        370      MOVP    A,@A
00CA 83        371      RET
372 ;DGPATS JE BAZE TABULKY SEGMENTOVYCH VZOREK PRO ZAKLADNI
373 ;ZNAKY (0-F)
374 ;PRO SPECIFICKE UZIVATELSKE APLIKACE TABULKA MUZE BYT MODIFIKOVANA
375 ;PODLE POTREBY S POUZITIM FORMATU SEGMENTU DISPLAY
376 ;FORMAT JE V NASEM PRIPADE STANDARTNI--P-G-F-E-D-C-B-A--
377 ;KDE -P-JE SVIT DESETINNE TECKY
378 ;
00CB          379 DGPATS EQU      $ AND OFFH
00CB 3F        380      DB      00111111B XOR SEGPOL
00CC 06        381      DB      00000110B XOR SEGPOL
00CD 58        382      DB      01011011B XOR SEGPOL
00CE 4F        383      DB      01001111B XOR SEGPOL
00CF 66        384      DB      01100110B XOR SEGPOL
00D0 6D        385      DB      01101101B XOR SEGPOL
00D1 7D        386      DB      01111101B XOR SEGPOL
00D2 07        387      DB      00000111B XOR SEGPOL
00D3 7F        388      DB      01111111B XOR SEGPOL
00D4 67        389      DB      01100111B XOR SEGPOL
00D5 77        390      DB      01110111B XOR SEGPOL
00D6 7C        391      DB      01111100B XOR SEGPOL
00D7 39        392      DB      00111001B XOR SEGPOL
00D8 5E        393      DB      01011110B XOR SEGPOL
00D9 79        394      DB      01111001B XOR SEGPOL
00DA 71        395      DB      01110001B XOR SEGPOL
396 ;
397 ;*****
398 ;
399 ;WDISP ZAPISUJE BINARNI VZORKU NYNI V ACCU DO DALSI ZNAKOVE POSICE
400 ;      DISPLAY (NEXTPL)
401 ;
00DB A9        402 WDISP:  MOV      PNTR1,A
00DC FF        403      MOV      A,NEXTPL
00DD 0337      404      ADD     A,#SEGMAP
00DF 29        405      XCH     A,PNTR1
00E0 A1        406      MOV      @PNTR1,A
00E1 EF05      407      DJNZ    NEXTPL,WDISP1
00E3 BF08      408      MOV      NEXTPL,#CHARNO
00E5 83        409 WDISP1: RET
410 ;
411 ;*****
412 ;RENTY SUBROUTINA K ZAPISU ACCU DO PRAVE KRAJNI POSICE DISPLAY
413 ;      S POSUNUTIM VSEHO DO LEVA O 1 POSICI
414 ;
00E6 B938      415 RENTY:  MOV      PNTR1,#SEGMAP + 1
00E8 BF08      416      MOV      NEXTPL,#CHARNO
00EA 21        417 RENT1:  XCH     A,@PNTR1
00EB 19        418      INC      PNTR1
00EC EFEA      419      DJNZ    NEXTPL,RENT1
00EE BF08      420      MOV      NEXTPL,#CHARNO ;JEDNOVA UKAZATELE NA LEVOU KRAJNI POSICI
421 ;
00F0 83        421      RET
422 ;
423 ;*****
424 ;
425 ;RDPADD DAVA DESETINNOU TECKU ZA POSLEDNI ZNAK NA DISPLAY
426 ;UPADD DAVA DESETINNOU TECKU KE ZNAKU KTERY JE V ACCU
427 ;
00F1 2301      428 RDPADD: MOV      A,#01H ;NASTAVENI INDEXU NA PRAVOU KRAJNI POSICI
00F3 0337      429 OPADD:  ADD     A,#SEGMAP ;JCITA DISPLAY REG.ADRESU PRO PRISL.POSICI
00F5 A9        430      MOV      PNTR1,A
00F6 F1        431      MOV      A,@PNTR1
00F7 D380      432      XRL     A,#80H
00F9 A1        433      MOV      @PNTR1,A
00FA 83        434      RET
435 ;
436 ;*****
437 ;      ORG      0100H ;POSUV NA DALSI STRANKU PAMETI PROGRAMU

```

```

438 ;HOLD SUBROUTINA VOLANA KDYZ JE STLACENE NEJAKE TLACITKO
439 ; A NEVRACI SE Z NI DOKUD NENI TOTO UVOLNENO
0100 D5 440 HOLD: SEL RB1
0101 FE 441 MOV A, LASTKY ;(LASTKY)=OFFH JAK NENI STLACENO NIC
0102 C5 442 SEL RBO
0103 37 443 CPL A
0104 9600 444 JNZ HOLD
0106 83 445 RET
446 ;
447 ;*****
448 ;
449 ;DELAY SUBROUTINA DAVA SPOZDENI ROVNE POCITU KOMPLETNICH REFRESH DISPLAY
450 ; V ZAVISLOSTI NA HODNOTE ZPOZDENI DANE ACCU PRED JEJIM VOLANIM
0107 B923 451 DELAY: MOV PNTR1, #RDELAY
0109 A1 452 MOV @PNTR1, A
010A F1 453 DELAY1: MOV A, @PNTR1
010B 960A 454 JNZ DELAY1
010D 83 455 RET
456 ;
457 ;*****
458 ;
011F 459 ORG 11FH ;ZACATEK UKAZEK
460 ;
461 ;PROGRAMY NA TETO STRANCE PAMETI PROGRAMU JSOU UKAZKOU POUZITI
462 ;NAVRZENHO HW A PRISLUSNYCH RUTIN SW
463 ;NESLOUZI SAMOZDREJME JAKO STANDARTNI FUNKCE TOMOTO PRIKLADU
464 ;APLIKACE JE UKAZKOU DEFINICE UZIVATELSKYCH RUTIN NA ZAKLADE
465 ;VYVOLANI FUNKCNIH TLACITEK
466 ;RUTINY JSOU VOLANY JELI 1 ZE 4 F=KEYS STLACENE
467 ;
468 ;
469 ;*****
470 ;
471 ; FUNCTN RUTINY K IMPLEMENTACI 1 ZE 4 DEMONSTRACNIH UKAZEK V ZAVISLOSTI
472 ; OD STLACENI 1 ZE 4 FUNKCNIH TLACITEK
473 ;
011F 1231 474 FUNCTN: JB0 FUNCT1
0121 322D 475 JB1 FUNCT2
0123 5229 476 JB2 FUNCT3
477 ;
0125 14F1 478 FUNCT4: CALL RDPADD
0127 0482 479 JMP ECHO
480 ;
0129 344D 481 FUNCT3: CALL TEST3
012B 0482 482 JMP ECHO
483 ;
012D 3443 484 FUNCT2: CALL TEST2
012F 0482 485 JMP ECHO
486 ;
0131 3435 487 FUNCT1: CALL TEST1
0133 0482 488 JMP ECHO
489 ;
490 ;*****
491 ;
492 ;TEST1 TENTO KODOVY SEGMENT VYPLNI DISPLAY REGISTRY V RAM DEKTIEMENTEM 8
0135 BFO8 493 TEST1: MOV NEXTPL, #CHARNO
0137 B808 494 MOV PNTR0, #CHARNO ;NASTAVENI 8 CYFROVE SMYCKY
0139 FF 495 TST11: MOV A, NEXTPL
013A 14C5 496 CALL ENCCACC
013C 14DB 497 CALL WDISP
013E E839 498 DJNZ PNTR0, TST11 ;KOPIROVANI HODNOTY DO DISPLAY REG.
0140 BFO8 499 MOV NEXTPL, #CHARNO
0142 83 500 RET
501 ;
502 ;*****
503 ;TEST2 ZAPISUJE SEGMENTOVOU VZORKU "AHOJ" DO DISPLAY REG.
504 ; CHVILI POCKA A POTOM VYMAZE DISPLAY
0143 B8C0 505 TEST2: MOV PNTR0, #AHOJ
0145 1486 506 CALL PRINT
0147 2364 507 MOV A, #100 ;CYKLUS DISPLAY 100 KRAT
0149 3407 508 CALL DELAY
014B 04A9 509 JMP CLEAR
510 ;
511 ;*****
512 ;
513 ;TEST3 SUBROUTINA VYPLNUJICI DISPLAY POMLCKAMI
514 ; SKACE DO SUBROUTINY 'CLEAR'
515 ; JAKMILE TLACITKO JE UVOLNENO
014D 2340 516 TEST3: MOV A, #01000000B XOR SEGPOL ;VZOREK '-'
014F 14AB 517 CALL FILL
0151 3400 518 CALL HOLD
0153 04A9 519 JMP CLEAR
520 ;
521 ;*****
522 ; TOBOLA JIRI 9.9.1988
523 ;*****
524 END

```

USER SYMBOLS			
ACCSA 0003	AHOJ 00C0	ASAVE 0002	BLANK 0000
CLR1 00AF	CURDIG 0007	DEBNCE 0004	CHARNO 0008
ENCCACC 00C5	ENCMASK 000F	FILL 00AB	CHRPOL 00FF
FUNCTN 011F	HOLD 0100	INIT 0068	CHRTB 0062
LEGND5 0099	NCOLS 0004	NEGLOG 00FF	DPADD 00F3
PINPUT 0009	PNTR0 0000	PNTR1 0001	DPAT3 00C8
RDELAY 0023	RDPADD 00F1	REFR1 0013	DPAT4 0125
SCAN 0029	SCAN1 002C	SCAN3 003F	DPAT5 012D
SEGPOL 0000	TST1 0135	TEST2 0143	DPAT6 012E
WDISP 00DB	WDISP1 00E5	TEST3 014D	DPAT7 012F
			DPAT8 0130
			DPAT9 0131
			DPAT10 0132
			DPAT11 0133
			DPAT12 0134
			DPAT13 0135
			DPAT14 0136
			DPAT15 0137
			DPAT16 0138
			DPAT17 0139
			DPAT18 013A
			DPAT19 013B
			DPAT20 013C
			DPAT21 013D
			DPAT22 013E
			DPAT23 013F
			DPAT24 0140
			DPAT25 0141
			DPAT26 0142
			DPAT27 0143
			DPAT28 0144
			DPAT29 0145
			DPAT30 0146
			DPAT31 0147
			DPAT32 0148
			DPAT33 0149
			DPAT34 014A
			DPAT35 014B
			DPAT36 014C
			DPAT37 014D
			DPAT38 014E
			DPAT39 014F
			DPAT40 0150
			DPAT41 0151
			DPAT42 0152
			DPAT43 0153
			DPAT44 0154
			DPAT45 0155
			DPAT46 0156
			DPAT47 0157
			DPAT48 0158
			DPAT49 0159
			DPAT50 015A
			DPAT51 015B
			DPAT52 015C
			DPAT53 015D
			DPAT54 015E
			DPAT55 015F
			DPAT56 0160
			DPAT57 0161
			DPAT58 0162
			DPAT59 0163
			DPAT60 0164
			DPAT61 0165
			DPAT62 0166
			DPAT63 0167
			DPAT64 0168
			DPAT65 0169
			DPAT66 016A
			DPAT67 016B
			DPAT68 016C
			DPAT69 016D
			DPAT70 016E
			DPAT71 016F
			DPAT72 0170
			DPAT73 0171
			DPAT74 0172
			DPAT75 0173
			DPAT76 0174
			DPAT77 0175
			DPAT78 0176
			DPAT79 0177
			DPAT80 0178
			DPAT81 0179
			DPAT82 017A
			DPAT83 017B
			DPAT84 017C
			DPAT85 017D
			DPAT86 017E
			DPAT87 017F
			DPAT88 0180
			DPAT89 0181
			DPAT90 0182
			DPAT91 0183
			DPAT92 0184
			DPAT93 0185
			DPAT94 0186
			DPAT95 0187
			DPAT96 0188
			DPAT97 0189
			DPAT98 018A
			DPAT99 018B
			DPAT100 018C

ASSEMBLY COMPLETE, NO ERRORS

# PROGRAM PRO MINIMALIZACI LOGICKÉ FUNKCE „BAJT“

Zbyněk Calaba, OK1 SZC, Krouzova 3039 143 00 Praha 4

Při návrhu kombinačních logických obvodů se konstruktér často setkává s potřebou vytvořit logickou síť určitých vlastností. Požadovaná funkce se zpravidla vyjadřuje pravdivostní tabulkou pro  $2^n$  stavů, kde  $n$  je počet vstupních logických proměnných. Pro určitý počet těchto stavů nabývá logická funkce hodnotu „1“ a pro ostatní „0“. Takovou funkci potom můžeme v triviálním případě vytvořit použitím dekodérů těch stavů, pro které funkce nabývá hodnoty „1“ a provést logický součet jejich výstupů.

Minimalizace logické funkce odpovídá na otázku, zda je vždy nutné při dekódování zahrnovat všechny vstupní proměnné, tj. zda by nebylo možné dva nebo více samostatných dekodérů nahradit jedním.

V praxi existují různé metody pro minimalizaci logických funkcí. Jejich pracnost však prudce roste s počtem vstupních proměnných. A člověk je tvor omylný a pohodlný...

V některých případech řeší situaci prostě tak, že k vytvoření funkce použije PROM. Přitom řešení s použitím „klasických“ hradel může vyhovět a navíc odpadnou problémy s programováním paměti.

K usnadnění tohoto rozhodování, pro řešení úlohy minimalizace logických funkcí, případně pro demonstrační účely v kroužcích a kursech, byl vytvořen tento program.

Praktické využití na mikropočítači SPECTRUM je omezeno aplikací na logickou funkci nejvýše osmi vstupních proměnných. Toto omezení je dáno typem „stroje“, neboť celá minimalizace je řešena v paměti omezené délky a v sestavě SPECTRA chybí prostředek pro efektivní práci se soubory.

V amatérské praxi je minimalizace i takové logické funkce přínosem. Kdo nevěří, ať to zkusí manuálně!

Hlavní program plní funkci editoru tabulek popisujících danou logickou funkci. Je napsán v jazyce BASIC-ROM SPECTRUM. Podprogram ve strojovém kódu zajistí velmi rychlý výpočet s výstupem minimalizace ve tvaru „součet součinů“ nebo „negace součtu součinů“. Je využit strojový kód Z-80, délka podprogramu je 1800 bajtů. Při práci programem se využívá celá zbývající paměť počítače.

Použití strojového kódu je nezbytné. V původní variantě byl algoritmus ověřován v jazyce BASIC. Nehledě na další omezení trval výpočet až 30 minut a to i při použití BASIC kompilátorů (BLAST, COLT).

## 1. Implementace programu

Připustme, že budoucí uživatel potřebuje především funkční program a že jej nezajímá, co se v paměti počítače děje. Potom se stává otisknutí výpisu v assembleru neúčelné. Uživatel by musel napsat do počítače podstatně více znaků a protože navíc programu nerozumí (proč také), nemůže ani posoudit, zda pomocí překladače vytvořil správný program, a to i v případě, kdy překladač nehlašuje žádné chyby. Pro takového uživatele je zapotřebí, aby mohl na základě údajů autora vytvořit rychle a přesně požadovaný blok údajů v paměti a uchovávat ho na kazetě.

K mnou použitému řešení mne inspiroval způsob používaný v časopise FUNKAMATEUR. Princip spočívá v tom, že je publikován výpis, jehož každý řádek je doplněn kontrolním číslem (je vypočítáno pro každý

řádek). Při vkládání údajů do paměti je kontrolní číslo opět vypočítáváno a kontrolováno se zadaným. Uživatel je ihned upozorněn na případnou chybu.

Domnívám se, že tento způsob otiskování programů by si zasloužil rozšíření i u nás. Ušetříme čas a také tiskové strany, kterých se nedostává. Navíc je takový způsob přístupný i „basicovským“ programátorům.

Ostatné, zdatní programátoři si zajisté najdou způsob, jak pořídit zpětný překlad do assembleru, pokud to ovšem potřebují. Nemluvě o programátorské etice, analýza a následná úprava cizího programu je časově náročná.

Ve Výpisu 4 je uveden program „HEXLOAD“, který umožní vytvoření bloku paměti na základě výpisu výše uvedených vlastností. Tento program je bezpečně použitelný pro vytváření bloků od adresy 6500 H výše.

Program „HEXDUMP“ (výpis 5) je určen pro kontrolní výpis paměti na displeji. V případě, že by se navrhovaný způsob prezentace programů ve strojovém kódu ujal, jeho úprava pro tiskárnu nebude činit potíže ani začátečníkům.

Oba programy si vytvářejí v řádce 1 REM strojový kód pro výpočet kontrolního čísla. Využívá se instrukce XOR.

Tyto programy doporučuji vytvořit předem a po ověření funkce uložit na kazetu ještě před zahájením dalších prací.

V paměti je nutno vytvořit dva bloky strojového kódu. Druhý blok je řídicí tabulka. K vytvoření doporučuji následující postup:

1. RESET systému (dojde k vynulování paměti).
2. Zadat postupně: CLEAR 32767, LOAD „HEXLOAD“, RUN.
3. Zadat adresu (8000) a uložit první blok (postačí do adresy 8370). Postupně se vždy zadává celý řádek, včetně kontrolního čísla podle šablony v dolní části obrazovky. Pro opravy můžete využívat běžné editační klávesy. V případě správně vložených dat je adresa automaticky zvětšena a ve vkládání můžete pokračovat. Nesprávně vložená data program signalizuje a řádek musíte napsat znova. Po dosažení konce bloku přerušíte program zadáním prázdného vstupního řetězce.
4. Zadáte RUN a celý postup budete opakovat pro druhý blok od adresy 8500.
5. Pořídíte bezpečnostní kopii podprogramu (příkazem SAVE „probajt“ CODE 32768, 1800) a provedete její verifikaci.
6. Zadáte NEW (smaže se oblast od 32767 dolů).
7. Vytvoříte hlavní program „BAJT“ podle výpisu 1. Pozor! Program se doporučuje

spouštět teprve v okamžiku, když je celý vytvořen! V žádném případě to nemůže být dříve, dokud není přítomen v paměti podprogram!

8. Příkazem GOTO 160 pořídíte potom úplnou kopii programu na kazetu.
9. Nyní je možno zahájit testování programu a opravy chyb vzniklých při zápisu programu v BASICu.
10. Konečnou verzi získáte použitím „auto-save“ na řádce 160 (viz bod 8).

### Poznámka:

Matematikové mohou dokázat, že způsob vytváření kontrolní čísel nezabrání všem možným omylům při vkládání podprogramu do paměti. V případě, že se program při vyvolání minimalizace zhroutí nebo se neočekávaně zachová, mohu doporučit jedině: program „HEXDUMP“ a kontrolu každého bajtu zvlášť.

## 2. Používání programu

Po zavedení programu je vypsáno základní menu:

- 1 – tabulku vytvořit,
- 2 – tabulku číst z magnetofonu,
- 3 – tabulku uložit na magnetofon,
- 4 – tabulku opravit,
- 5 – tabulku zobrazit,
- 6 – tabulku vytisknout,
- 7 – tabulku minimalizovat,
- 8 – tabulku invertovat.

Řádky 3 až 8 jsou zobrazovány pouze tehdy, jestliže program již nějakou tabulku zná.

Program pracuje se soubory (typ 3 – CODE) délky 256 bajtů, jejichž jméno je tvořeno šesti volitelnými znaky a sufixem „TAB“.

Použití tlačítka BREAK přeruší program, s výjimkou případu, kdy pracuje podprogram minimalizace. Pro návrat k menu bez ztráty dat použijte povel GOTO 22. Příkaz RUN také spustí program, avšak způsobí vynulování tabulky.

Hodnoty jsou do programu zadávány v hexadecimálním tvaru. Zadávané hodnoty (řetězce) jsou vždy interpretovány na úrovni posledních dvou pravých znaků, zbytek řetězce je ignorován. Vstup prázdného řetězce je ekvivalentní vstupu „0“ nebo je využit pro ukončení funkce (program vždy upozorní).

### Vytvoření tabulky

Tabulka v paměti je vynulována a v následujícím dialogu jsou zadávány termíny, tj. stavby, pro které nabývá výstupní logická funkce

hodnotu „1“. Celý proces vkládání ukončíte vstupem řetězce nulové délky (stiskem samotné klávesy ENTER). Poté je automaticky zobrazena tabulka, kterou jste vytvořili a jste dotázáni, zda souhlasíte se zadáním (s jejím obsahem).

Pokud odpovíte „A“, přejde program zpět k hlavnímu menu. Odpověď „N“ vyvolá dotaz, zda si přejete tabulku jen opravit („O“) nebo vytvořit znovu („Z“). Rozdíl spočívá v tom, že na pokyn „O“ přejde program k funkci 4, pokyn „Z“ způsobí vynulování tabulky (návrat k počátku funkce 1). Jiné odpovědi program odmítá.

Uvedený postup je vhodný pro vytvoření tabulky, která je málo obsazena hodnotou „1“. Tabulku s převládajícím počtem jedniček získáte snáze tímto postupem:

- bezprostředně po volbě 1 ukončíte práci (ENTER),
- odsouhlasíte nulovou tabulku („A“),
- po zobrazení menu provedete volbu 8,
- použijete volbu 4 – opravit tabulku.

### Čtení tabulky z magnetofonu

Po této volbě jste dotázáni na název tabulky, kterou chcete číst. Délka názvu je zkontrolována. Sufix „TAB“ nezačínáte, je doplňován automaticky. Potom zbývá běžná obsluha počítače při čtení souboru. Po úspěšném čtení se program vrací k hlavnímu menu.

V případě, že zadáte prázdný řetězec místo názvu, je nahrán prvý soubor typu 3, ovšem jen za předpokladu, že má shodné parametry jako soubory vytvářené programem „BAJT“. Jinak nastane chyba a je nutno program znovu spustit.

### Zápis tabulky na magnetofon

Jste požádáni o název tabulky (opět nezačínáte sufix „TAB“). Vstup prázdného řetězce program odmítá, stejně jako název delší než šest znaků. Další postup je shodný s postupem pro práci se soubory. Po nahrání máte možnost volbou „A“ verifikovat uložená data. Jiná volba, stejně jako úspěšná verifikace dat, vede k návratu k hlavnímu menu.

### Oprava tabulky

Tabulka je opravována po jednotlivých prvcích, jejichž pozici je vždy nutno zadat. Zadáváte tedy nejdříve pozici, potom je znázorněna stará hodnota a potom jste požádáni o vstup hodnoty nové.

Vstup prázdného řetězce při dotazu na pozici způsobí zobrazení tabulky a žádost o její odsouhlasení. Pozor! Odpověď „Z“ vede nenávratně ke ztrátě dat a funkci 1, tj. znovuvytvoření tabulky! Pokud si přejete pokračovat v opravách, je nutno zadat „N“ a na následující otázku odpovědět znakem „O“.

### Zobrazení tabulky na displeji

Tato volba způsobí, že tabulka je zobrazena ve tvaru mapy, kdy pořadí indexů v řádcích a sloupcích je změněno proti přirozené posloupnosti hexadecimálních čísel. Tento způsob poskytuje konkrétnější grafickou představu o povaze logické funkce a možnostech její minimalizace.

### Vytisknutí tabulky na tiskárnu

Výstup je zajištěn příkazem LPRINT. Máte možnost zadat pro tabulku libovolný nadpis. Vzor výpisu je uveden v kontrolním příkladu na obr. 1.

Ti uživatelé, kteří nemají k dispozici originální tiskárnu, musí nahrát před spuštěním programu do paměti obslužný podprogram a provést inicializaci. Doporučuji jeho umístění v rozmezí adres 5B00–5BFF, rozhodně však do adresy 7FFF.

### LOGICKÁ FUNKCE (OR-AND):

```
{
  (/A/B/E)
+(/A/B/C/D)
+( A B C D)
}
```

Tabulka k předcházející logické funkci

	0	1	3	2	6	7	5	4	C	D	F	E	A	B	9	8
0:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1:	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
3:	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
2:	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
6:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
E:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### LOGICKÁ FUNKCE (NOR-AND):

```
{
  (/A/B/E)
+(/A/B/C/D)
+( A B C D)
}
```

Obr. 1. Výpisy kontrolního příkladu (921–1)

### Minimalizace tabulky

Volbou je spuštěn podprogram minimalizace tabulky logické funkce. Po dobu řešení je zakázáno přerušení. Doba nutná k řešení závisí na obsahu tabulky. Akusticky jste upozorněni, jestliže program nalezl řešení. Po stisknutí libovolné klávesy je řešení vypisováno. Tvar výpisu bude rozebrán dále. Akustická signalizace je pozůstatkem z dřívější doby, je možné ji vypustit. Po ukončení výpisu máte možnost požádat o jeho zopakování (vhodné u delších výpisů).

### Invertování tabulky

Volba způsobí záměnu všech prvků „0“ tabulky za „1“ a obráceně.

## 3. Kontrolní příklad

Pro ověření základních funkcí vám poslouží následující kontrolní případ. Máme vytvořit kombinační obvod, pro který platí tabulka uvedená uprostřed obr. 1. Po spuštění programu zvolte „1“ a potom zadávejte:

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17
20 21 22 23 24 25 26 27
30 31 32 33 34 35 36 37
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
„prázdný vstup“.
```

Připomínám, že na pořadí zadávaných hodnot nezáleží, zde prezentovaný tvar vám zadání pouze usnadní. Pokud jste se zmýlili, máte možnost již popsaným postupem tabulku opravit. Na konci celého Vašeho snažení by měla být tabulka zmíněná v úvodu.

Volbou „3“ uložíte tabulku na pásku. Ověřte i verifikaci dat.

Smazáním tabulky (volba „1“ a ihned ENTER) obdržíte nulovou tabulku. Pomocí volby „2“ vyzkoušíte čtení z magnetofonu. Musíte obdržet původně vytvořenou tabulku.

Volbou „4“ vyzkoušíte opravu tabulky (pokud jste tak již neučinili v kroku vytváře-

ni). Zadáte např. AA 1 AA 0. Tabulka je opět původní.

Volbou „8“ ověřte inverzi tabulky. Aby bylo dosaženo stejných výsledků jako na obrázku, proveďte inverzi ještě jednou.

Pokud máte připojenou tiskárnu, můžete ověřit tisk tabulky volbou „6“. Její zobrazení (volba „5“) bylo ověřeno již v mnoha předchozích krocích.

Volba „7“ způsobí, že po jisté době bude na váš pokyn zobrazeno řešení ve tvaru uvedeném v horní části obrázku 1.

Nakonec postupnou volbou „8“ a „7“ obdržíte výsledek v případě invertované tabulky (obr. 1 dole).

### Poznámky k tvaru výpisu logické funkce:

Jednotlivé vstupní proměnné jsou vyjádřeny znaky „A“ až „H“. Znak „/“ před proměnnou vyznačuje negaci proměnné například „/A“ znamená „ne A“, „A non“.

Jednotlivé dílčí součiny jsou uzavřeny v kulatých závorkách, složené závorky uzavírají celou funkci.

Jak již bylo uvedeno, program se snaží šetřit čas a proto v závislosti na rozsahu zadání volí dvě vyjádření logické funkce. Druhý typ výpisu se bude někomu zdát neobvyklý. Využitím pravidel logiky je možno výsledek upravit na aplikaci konkrétně použitých logických prvků. Já jsem si nechtěl komplikovat situaci rozšířením programu pro znázorňování různých variant, za hlavní přínos považuji především zpracování podprogramu minimalizace.

Pokud bude vstupních proměnných méně než osm, zadejte tabulku v menším rozsahu termů a ve výpisu nepřihlížejte k nevyužitým proměnným.

Závěrem přeji všem uživatelům mnoho spokojenosti při práci s tímto programem.

### Výpis 1. Hlavní program (921–V1)

```
1 REM -----
2 REM *BAJT* minimalizace GMC
3 REM -----
4 LET pgm=32768: POKE 23658,8
5 DEF FN a(i)=PEEK (pgm+i)+25
6*PEEK (pgm+i+1)
6 LET main=FN a(0)
7 LET t=FN a(2)
8 RESTORE 14: DIM e(16)
9 DIM ex(16): DIM qx(8)
10 FOR i=1 TO 16
11 READ e(i),ex(i): NEXT i
12 FOR i=1 TO 8: READ qx(i)
13 NEXT i
14 DATA 0,"0",1,"1",3,"3",2,"2",
15 DATA 6,"6",7,"7",5,"5",4,"4",
16 DATA 12,"C",13,"D",15,"F",1
17 DATA 4,"E",10,"A",11,"B",
18 DATA 9,"9",8,"8",14,"B",13,"C",
19 DATA 1,"D",16,"E",17,"C",18,"H"
17 LET CTRL=2: DIM Zx(8,13)
18 FOR I=1 TO 8: READ Zx(I)
19 NEXT I
20 DATA "VYTVOŘIT","CIST Z MG
F","ULOŽIT NA MGF","OPRAVIT","ZO
BRAZIT","VYTISKNOUT"
21 DATA "MINIMALIZOVAT","INVER
TOVAT"
22 CLS: PRINT " BRIGHT 1;" **
***** BAJT *****
23 PRINT AT 2,3;"PROGRAM PRACE
S TABULKOU"
24 PRINT BRIGHT 1; " Zbynek
CALABA Praha 4 - 1988 "
25 PLOT 0,168: DRAW 255,0
26 DRAW 0,-25: DRAW -255,0
27 DRAW 0,25
28 PRINT "'TAB 1; BRIGHT 1;"M
OZNOSTI PROGRAMU:" BRIGHT 0;" "
```

```

29 FOR I=1 TO CTRL
30 PRINT " "; I; " - TABULKU ";
Z(I); NEXT I
31 NEXT I
32 PLOT 0,120: DRAW 255,0
33 DRAW 0,-90: DRAW -255,0
34 DRAW 0,90
35 PRINT #0; FLASH 1; BRIGHT 1
;"VOLTE, prosim!"
36 BEEP .1,12
37 LET ix=INKEYX
38 IF ix="" THEN GO TO 37
39 LET i=CODE ix
40 IF i>56 OR i<48 THEN GO TO
37
41 IF ix="1" THEN GO SUB 51:
GO TO 22
42 IF ix="2" THEN GO TO 106
43 IF VAL ix>ctrl THEN GO TO
37
44 IF ix="3" THEN GO TO 126
45 IF ix="4" THEN GO SUB 73:
GO TO 22
46 IF ix="5" THEN GO TO 89
47 IF ix="6" THEN GO SUB 137:
GO TO 22
48 IF ix="7" THEN GO TO 171
49 IF ix="8" THEN GO TO 208
50 GO TO 37
51 LET x=1: GO SUB 214
52 RANDOMIZE USR 33557: REM nu
lovani
53 PRINT "Pro popis tabulky z
adavejte term""(pozici '1' v ta
bulce)."
54 PRINT "Samotny ENTER ukonc
i vstup dat."
55 BEEP .05,12
56 LET jx="Prosim term"
57 GO SUB 113
58 IF i>255 THEN GO TO 63
59 LET m=PEEK (t+i)
60 IF m = 48 THEN POKE (t+i),
49: GO TO 55
61 PRINT #0;"DUPLICITA!"
62 BEEP .5,1: GO TO 55
63 GO SUB 90
64 BEEP .1,5: BEEP .1,9
65 BEEP .1,12
66 INPUT "V poradku (A/N)?",ix
67 IF ix="A" THEN LET ctrl=8:
GO TO 22
68 IF ix<>"N" THEN BEEP .5,1:
GO TO 64
69 BEEP .1,12
70 INPUT "Opravit - znova (O/Z
)?",ix
71 IF ix="Z" THEN GO TO 51
72 IF ix<>"O" THEN BEEP .5,1:
GO TO 69
73 LET X=4: GO SUB 214
74 PRINT "Samotny ENTER ukonc
i vstup dat."
75 PRINT AT 19,0;" "
76 BEEP .05,12
77 LET jx="Pozice v tabulce"
78 GO SUB 113: LET j=i
79 IF i>255 THEN GO TO 63
80 LET m=PEEK (t+j)
81 PRINT AT 19,0;"TAB(" ;ix;")=
";CHR$ m
82 BEEP .05,12
83 LET jx="Nova hodnota"
84 GO SUB 113
85 IF i=999 THEN, LET i=0
86 POKE (t+j),48+i
87 IF i<0 OR i>1 THEN BEEP .5
,1: GO TO 83
88 GO TO 75
89 GO SUB 90: GO TO 123
90 LET X=5: GO SUB 214
91 PRINT " "; FOR I=1 TO 16
92 PRINT BRIGHT 1;EX(I);
93 NEXT I
94 PRINT " FOR I=0 TO 15
95 PRINT BRIGHT 1;EX(I+1);
96 FOR J=0 TO 15
97 LET M=PEEK (t+e(J+1)+16*e(I
+1))
98 PRINT CHR$ m;
99 NEXT J: PRINT : NEXT I

```

```

100 RETURN
101 PRINT #0;"Stop MGF, prosim!
"; PAUSE 255: RETURN
102 INPUT "Jmeno tabulky (max.
6 znaku): ",ix
103 IF LEN ix>6 THEN BEEP .5,1
: GO TO 102
104 IF LEN ix=0 THEN RETURN
105 LET ix=ix+"TAB": RETURN
106 LET X=2: GO SUB 214
107 GO SUB 102
108 PRINT AT 3,0;"Jmeno: ";ix
109 PRINT #0;"Prosim magnetofon
!"
110 LOAD ixCODE t,256
111 GO SUB 101
112 LET ctrl=8: GO TO 22
113 INPUT (jx+" (hex):"),ix
114 IF LEN ix=0 THEN LET i=999
: RETURN
115 LET ix="0"+ix
116 IF LEN ix>2 THEN LET ix=ix
(2 TO ): GO TO 116
117 LET k=CODE ix(1): LET l=COD
E ix(2)
118 LET e=k-(48+(k>57)*39)
119 LET k=e+((e<0)*32)
120 LET e=l-(48+(l>57)*39)
121 LET l=e+((e<0)*32)
122 LET i=16*k+l: RETURN
123 BEEP .1,12
124 INPUT "Prosim ENTER!";ix
125 GO TO 22
126 LET X=3: GO SUB 214
127 GO SUB 102
128 IF LEN ix<>0 THEN GO TO 13
1
129 PRINT #0;"Tabulka musi mit
jmeno!"
130 BEEP .5,1: GO TO 127
131 PRINT "Jmeno: ";ix
132 SAVE ixCODE t,256
133 INPUT "Verifikovat (A)?",jx
134 IF jx<>"A" THEN GO TO 22
135 PRINT "Verifikace dat ...":
VERIFY ixCODE t,256
136 PRINT "Data o.k.": GO TO 12
3
137 LET X=6: GO SUB 214
138 GO SUB 153
139 LPRINT " ";
140 FOR I=1 TO 16:
141 LPRINT EX(I); " ";
142 NEXT I: LPRINT
143 LPRINT " ";
144 FOR I=0 TO 15: LPRINT "==",
145 NEXT I: LPRINT
146 FOR I=0 TO 15
147 LPRINT EX(I+1); " ";
148 FOR J=0 TO 15
149 LET M=PEEK (t+e(J+1)+16*e(I
+1))
150 LPRINT m-48; " ";
151 NEXT J: LPRINT : NEXT I
152 RETURN
153 INPUT "Nadpis sestavy: ";jx
154 LPRINT jx: LPRINT
155 RETURN
156 CLEAR 32767
157 PRINT "AUTOSAVE-GO TO 160"
158 LOAD "probajt"CODE 32768,18
00: RUN
159 REM autosave programu
160 CLEAR : SAVE "BAJT" LINE 15
6
161 SAVE "probajt"CODE 32768,18
00
162 STOP : GO TO 1
163 PRINT "JDE O TRIVIALNI RE
SENI"
164 BEEP .05,5: BEEP .5,9
165 BEEP .05,5
166 PAUSE 255: GO TO 22
167 BEEP 1,1
168 PRINT "Zjistena chyba pr
ogramu."
169 PRINT "Doporucuji nahrat pr
ogram znovu."
170 PRINT "a pred reklamaci opa
kovat.": PAUSE 0: GO TO 22

```

```

171 LET X=7: GO SUB 214
172 PRINT "Cekajte, prosim!"
;"Doba reseni zavisi na slozito
sti"
173 PRINT "tabulky. Reseni ohla
sim!"
174 RANDOMIZE USR main
175 LET od=FN a(4)
176 LET do=FN a(6)
177 LET rc=PEEK (pgm+8)
178 IF rc=1 THEN GO TO 163
179 IF rc=3 THEN GO TO 181
180 IF rc<>0 THEN GO TO 167
181 CLS : PRINT "FLASH 1; BRI
GHT 1;"R E S E N I !"
182 PRINT BRIGHT 1;"Stisknete
klavesu."
183 BEEP .05,5: BEEP .05,9
184 IF INKEYX="" THEN GO TO 18
3
185 IF rc=3 THEN GO TO 201
186 CLS : PRINT BRIGHT 1;"LOGI
CKA FUNKCE (OR-AND):"; PRINT
187 PRINT "{": GO SUB 191: PRI
NT "}"
188 GO SUB 211
189 IF ix="A" THEN GO TO 186
190 GO TO 22
191 LET i=od: PRINT " (" ;
192 LET j=PEEK (i+1)
193 FOR k=1 TO 8: LET j=PEEK (i
+k)
194 IF j=45 THEN GO TO 197
195 IF j=48 THEN PRINT "/";qx(
k); GO TO 197
196 IF j=49 THEN PRINT " ";qx(
k);
197 NEXT k: PRINT ")"
198 LET i=i+10
199 IF i<>do THEN PRINT "+(" ;
GO TO 193
200 RETURN
201 CLS : PRINT BRIGHT 1;"LOGI
CKA FUNKCE (NOR-AND):";
202 PRINT "PRINT BRIGHT 1;"/{
"
203 GO SUB 191
204 PRINT BRIGHT 1;" }": BEEP
.05,12
205 GO SUB 211
206 IF ix="A" THEN GO TO 201
207 RANDOMIZE USR 33572: GO TO
22
208 LET X=8: GO SUB 214
209 RANDOMIZE USR 33572: PAUSE
10'
210 GO TO 89
211 INPUT "Opakovat vypis resen
i (A)? ";ix
212 RETURN
213 BEEP .1,12: CLS : RETURN
214 GO SUB 213: PRINT BRIGHT 1
;"TABULKU ";zx(x)
215 RETURN

```

## Výpis 2. Podprogram (921-V2)

Vypis s kontrolou

8000	09	80	00	84	38	88	4F	88	79
8008	03	F3	FD	22	00	88	21	00	86
8010	84	7E	2C	28	05	BE	20	07	62
8018	18	F8	3E	01	C3	5E	82	AF	6F
8020	32	08	80	21	00	84	01	00	1E
8028	00	3E	31	BE	20	01	04	23	B7
8030	0D	20	F8	3E	80	B8	30	1C	FF
8038	3E	03	32	08	80	21	00	84	22
8040	01	31	30	7E	B8	20	03	71	94
8048	18	05	B9	C2	57	82	70	23	E0
8050	2C	2D	20	EF	21	00	00	22	CD
8058	37	88	22	39	88	3E	01	B8	AB
8060	28	B8	AF	32	00	87	01	FF	74
8068	00	11	01	87	21	00	87	ED	DC
8070	80	21	38	88	22	04	80	22	A6
8078	06	80	11	00	84	1A	FE	31	C6
8080	20	09	21	00	86	68	6E	26	AD
8088	87	36	FF	13	1C	1D	20	ED	91
8090	11	00	87	2A	06	80	1A	3C	1C
8098	20	26	D5	16	85	1A	01	08	53
80A0	00	E5	23	07	F5	38	04	3E	36
80A8	30	18	03	3E	31	04	77	23	74



```

8080 F1 0D 20 EF AF 77 E1 70.7A
8088 01 0A 00 09 22 06 80 D1.77
80C0 13 1C 1D 20 D1 CD C6 82.6A
80C8 FE 00 C2 57 82 2A 06 80.45
80D0 22 08 88 22 06 88 3E 08.38
80D8 F5 FB 3E 7F DB FE 1F 38.4D
80E0 03 C3 56 82 F3 ED 4B 06.47
80E8 80 2A 04 80 CD 67 82 CA.CC
80F0 56 82 3E 08 11 13 88 ED.85
80F8 53 02 88 F5 2A 02 88 7E.F2
8100 23 B6 CA FD 81 23 7E 23.5D
8108 B6 CA FD 81 DD 2A 02 88.7D
8110 DD 6E 00 DD 66 01 E5 DD.31
8118 2A 02 88 DD 5E 02 DD 56.AA
8120 03 D5 E5 D5 E5 D5 AF 08.71
8128 AF 32 0A 88 11 08 88 D5.58
8130 23 DD E1 01 08 00 ED B0.4B
8138 0E 08 D1 E1 E5 D5 EB DD.30
8140 2B 13 23 DD 23 1A BE 28.69
8148 1E 08 3C FE 02 28 21 08.DC
8150 15 2D 28 18 3E 2D BE 28.66
8158 05 DD 77 00 18 0E DD 7E.1A
8160 00 FE 31 20 07 3A 0A 88.50
8168 3C 32 0A 88 00 20 D2 08.7B
8170 DD E1 FD E1 FE 01 20 53.AC
8178 3E FF DD 77 09 FD 77 09.E1
8180 E5 D5 C5 ED 4B 08 88 2A.F9
8188 06 88 CD 67 82 28 23 22.8F
8190 04 88 0E 09 11 0A 88 1A.02
8198 BE 20 07 13 23 DD 20 F7.73
81A0 18 26 01 0A 00 2A 04 88.93
81A8 09 ED 4B 08 88 CD 67 82.07
81B0 20 DD 2A 08 88 E5 01 0A.B9
81B8 00 09 22 08 88 D1 01 09.72
81C0 00 21 0A 88 ED B0 AF 12.43
81C8 C1 D1 E1 D1 E1 D1 EB DD.26
81D0 2A 02 88 DD 46 15 DD 4E.BD
81D8 14 CD 67 82 28 08 01 0A.17
81E0 00 09 EB C3 21 81 E1 DD.BD
81E8 2A 02 88 DD 46 13 DD 4E.B8
81F0 12 CD 67 82 28 07 01 0A.1E
81F8 00 09 C3 16 81 CD 8F 82.9D
8200 F1 2A 02 88 23 23 22 02.71
8208 88 3D C2 FB 80 CD 8F 82.CC
8210 2A 06 80 ED 5B 04 80 37.A9
8218 3F ED 52 E5 C1 2A 08 88.0E
8220 ED 5B 06 88 37 3F ED 52.8F
8228 CD 67 82 20 18 2A 04 80.BE
8230 ED 5B 06 88 ED 4B 06 80.18
8238 1A BE 20 09 23 13 CD 67.17
8240 82 28 18 18 F3 CD 6D 82.7B
8248 CD C6 82 FE 00 20 07 F1.A1
8250 3D C2 D8 80 18 01 E1 3E.61
8258 02 18 03 F1 18 03 32 08.C9
8260 80 FD 2A 00 88 FB C9 7C.91
8268 B8 C0 7D B9 C9 ED 4B 06.D5
8270 88 C5 ED 5B 04 80 2A 08.5D
8278 88 37 3F ED 42 E5 C1 E1.EA
8280 ED B0 ED 53 06 88 ED 53.D3
8288 06 80 ED 53 08 88 C9 DD.AC
8290 2A 02 88 DD 6E 00 DD 66.A8
8298 01 7D B4 C8 E5 FD E1 3E.C7
82A0 FF FD BE 09 28 10 ED 5B.3B
82A8 08 88 01 0A 00 ED B0 ED.3B
82B0 53 08 88 FD E5 E1 DD 46.B1
82B8 13 DD 4E 12 CD 67 82 C8.72
82C0 11 0A 00 ED B0 2A 04 80.D4
82C8 DD 21 11 88 FD 21 23 88.12
82D0 21 13 88 AF 77 E5 D1 13.45
82D8 01 24 00 ED B0 2A 04 80.D6
82E0 11 0A 00 ED 4B 06 80 3E.05
82E8 FF 08 08 BE 28 16 DD 23.81
82F0 DD 23 FD 23 FD 23 3C FE.3C
82F8 09 28 17 BE 20 F0 DD 75.F0
8300 00 DD 74 01 FD 75 00 FD.DD
8308 74 01 08 19 CD 67 82 20.6C
8310 D9 AF FD E1 C9 3E 30 21.8C
8318 00 84 E5 D1 13 77 01 FF.2A
8320 00 ED B0 C9 21 00 84 01.30
8328 31 30 7E B8 20 03 71 18.8D
8330 01 70 23 2C 2D 20 F3 C9.49
8338 00 00 00 00 00 00 00 00.00
8340 00 00 00 00 00 00 00 00.00
8348 00 00 00 00 00 00 00 00.00
8350 00 00 00 00 00 00 00 00.00
8358 00 00 00 00 00 00 00 00.00
8360 00 CD 8C 82 00 00 2A 06.EF
8368 80 C3 10 82 00 00 00 00.D1
8370 00 00 00 00 00 00 00 00.00
8378 00 00 00 00 00 00 00 00.00
8380 00 00 00 00 00 00 00 00.00
8388 00 00 00 00 00 00 00 00.00

```

```

8390 00 00 00 00 00 00 00 00.00
8398 00 00 00 00 00 00 00 00.00
83A0 00 00 00 00 00 00 00 00.00
83A8 00 00 00 00 00 00 00 00.00
83B0 00 00 00 00 00 00 00 00.00
83B8 00 00 00 00 00 00 00 00.00
83C0 00 00 00 00 00 00 00 00.00
83C8 00 00 00 00 00 00 00 00.00
83D0 00 00 00 00 00 00 00 00.00
83D8 00 00 00 00 00 00 00 00.00
83E0 00 00 00 00 00 00 00 00.00
83E8 00 00 00 00 00 00 00 00.00
83F0 00 00 00 00 00 00 00 00.00
83F8 00 00 00 00 00 00 00 00.00

```

### Výpis 3. Tabulka (921-V3)

#### Vypis s kontrolou

```

8500 00 01 02 04 08 10 20 40.7F
8508 80 03 05 06 09 0A 0C 11.9E
8510 12 14 18 21 22 24 28 30.21
8518 41 42 44 48 50 60 81 82.3C
8520 84 88 90 A0 C0 07 0B 0D.FD
8528 0E 13 15 16 19 1A 1C 23.22
8530 25 26 29 2A 2C 31 32 34.1B
8538 38 43 45 46 49 4A 4C 51.66
8540 52 54 58 61 62 64 68 70.21
8548 83 85 86 89 8A 8C 91 92.0C
8550 94 98 A1 A2 A4 A8 B0 C1.72
8558 C2 C4 C8 D0 E0 F0 17 1B.FD
8560 1D 1E 27 28 2D 2E 33 35.0A
8568 36 39 3A 3C 47 48 4D 4E.06
8570 53 55 56 59 5A 5C 63 65.09
8578 66 69 6A 6C 71 72 74 78.06
8580 87 88 8D 8E 93 95 96 99.06
8588 9A 9C A3 A5 A6 A9 AA AC.09
8590 B1 B2 B4 B8 C3 C5 C6 C9.06
8598 CA CC D1 D2 D4 D8 E1 E2.0A
85A0 E4 E8 F0 F1 F2 F3 F8 3D.FD
85A8 3E 4F 57 58 5D 5E 67 68.72
85B0 6D 6E 73 75 76 79 7A 7C.0C
85B8 8F 97 9B 9D 9E A7 AB AD.21
85C0 AE B3 B5 B6 B9 BA BC C7.66
85C8 CB CD CE D3 D5 D6 D9 DA.1B
85D0 DC E3 E5 E6 E9 EA EC F1.22
85D8 F2 F4 F8 F9 5F 6F 77 78.FD
85E0 7D 7E 9F AF B7 BB BD BE.3C
85E8 CF D7 DB DD DE E7 EB ED.21
85F0 EF F3 F5 F6 F9 FA FC 7F.9E
85F8 BF DF EF F7 FB FD FE FF.7F
8600 00 01 02 09 03 0A 0B 25.2D
8608 04 0C 0D 26 0E 27 28 5D.7F
8610 05 0F 10 29 11 2A 2B 5E.7D
8618 12 2C 2D 2F 2E 60 61 A3.C0
8620 06 13 14 2F 15 30 31 62.58
8628 16 32 33 63 34 64 65 A4.E5
8630 17 35 36 66 37 67 68 A5.EF
8638 38 69 6A 66 6B A7 AB DB.22
8640 07 18 19 39 1A 3A 3B 6C.48
8648 1B 3C 3D 6D 3E 6E 6F A9.E1
8650 1C 3F 40 70 41 71 72 AA.FB
8658 42 73 74 AB 75 AC AD DC.46
8660 1D 43 44 76 45 77 78 AE.88
8668 46 79 7A AF 7B B0 B1 DD.4D
8670 47 7C 7D B2 7E B3 BA DE.53
8678 7F B5 B6 DF B7 E0 E1 F7.E2
8680 08 1E 1F 48 20 49 4A 80.E2
8688 21 4B 4C 81 4D 82 83 B8.53
8690 22 4E 4F 84 50 85 86 B9.4D
8698 51 87 88 BA 89 BB BC ED.88
86A0 23 52 53 8A 54 8B 8C BE.46
86A8 55 8D 8E BE 8F BF C0 E3.FB
86B0 56 90 91 C1 92 C2 C3 E4.E1
86B8 93 C4 C5 E5 C6 E6 E7 F8.48
86C0 24 57 58 94 59 95 96 C7.22
86C8 5A 97 98 C8 99 CA EB.EF
86D0 5B 9A 9B C9 CC CD E9.E5
86D8 9D CE CF EA D0 EB EC F9.58
86E0 5C 9E 9F D1 A0 D2 D3 ED.C0
86E8 A1 DA D5 EE D6 EF F0 FA.7D
86F0 A2 D7 D8 F1 D9 F2 F3 FB.7F
86F8 DA F4 F5 FC F6 FD FE FF.2D

```

### Výpis 4. Program „HEXLOAD“ (921-V4)

```

1 REM *****konec
2 FOR i=23760 TO 23776
3 READ a: POKE i,a: NEXT i

```

```

4 DATA 175,50,224,92,201,58
5 DATA 224,92,79,58,223,92
6 DATA 169,24,242,0,0
7 POKE 23658,8
8 POKE 23609,120
9 DIM B(8)
10 INPUT "ADRESA (HEX).",AX
11 IF LEN AX<4 THEN LET AX=""
  "+AX: GO TO 11
12 IF LEN AX>4 THEN LET AX=AX
  (2 TO ): GO TO 12
13 LET I=1: GO SUB 48
14 IF D<0 OR D>255 THEN GO TO
  10
15 LET A=256*D
16 LET I=3: GO SUB 48
17 IF D<0 OR D>255 THEN GO TO
  10
18 LET A=A+D
19 CLS: PRINT "ADRESA: ";AX;"
  =" ;A
20 LET A=A-1
21 INPUT "-----DATA-----"
  "-AX: LET K=0
22 IF LEN AX=0 THEN STOP
23 IF LEN AX<>19 THEN GO TO 4
  5
24 IF AX(17)<>" " THEN GO TO
  45
25 FOR I=1 TO LEN AX-4 STEP 2
26 GO SUB 48
27 IF D<0 OR D>255 THEN GO TO
  45
28 LET K=K+1: LET B(K)=D
29 NEXT I
30 REM KONTROLNI SUMA
31 LET I=18: GO SUB 48
32 IF D<0 OR D>255 THEN GO TO
  45
33 REM TEST
34 RANDOMIZE USR 23760
35 FOR I=1 TO 8
36 POKE 23775,B(I)
37 RANDOMIZE USR 23765
38 NEXT I: LET I=PEEK 23776
39 IF I<>D THEN GO TO 45
40 REM ULOZIT
41 FOR I=1 TO 8
42 POKE (A+I),B(I)
43 NEXT I: LET A=A+9
44 GO SUB 51: GO TO 20
45 PRINT AX;"...CHYBA!"
46 BEEP 1,1: GO TO 21
47 REM CTENI HEX
48 LET D=((CODE AX(I)-(48+(COD
  E AX(I)>57)*7)))*16
49 LET D=D+((CODE AX(I+1)-(48+
  (CODE AX(I+1)>57)*7)))
50 RETURN
51 CLS: PRINT "ADRESA: ";
52 LET D=INT (A/256)
53 GO SUB 57
54 LET D=A-256*D
55 GO SUB 57
56 PRINT " = ";A: RETURN
57 LET AX=""
58 LET A1=INT (D/16)
59 LET A2=D-16*A1
60 LET AX=CHR* (48+A1+((A1>9)
  *7))
61 LET AX=AX+CHR* (48+A2+((A2
  >9)*7))
62 PRINT AX: RETURN

```

### Výpis 5. Program „HEXDUMP“ (921-V5)

```

1 REM *****konec
2 POKE 23658,8
3 POKE 23609,120
4 PRINT "Vypis s kontrolou"
5 FOR I=23760 TO 23776
6 READ A: POKE I,A: NEXT I
7 DATA 175,50,224,92,201,58
8 DATA 224,92,79,58,223,92
9 DATA 169,24,242,0,0
10 POKE 23658,8: POKE 23609,1
11 INPUT "START(HEX).",AX
12 GO SUB 40
13 IF A<0 THEN BEEP 1,1: GO T
  0 11

```

```

14 LET Q1=A
15 INPUT "STOP (HEX).":A*
16 GO SUB 40
17 IF A<0 THEN BEEP 1.1: GO T
0 15
18 LET Q2=A
19 FOR I=Q1 TO Q2 STEP 8
20 RANDOMIZE USR 23760
21 LET A=INT (I/256)
22 GO SUB 33
23 LET A=I-256*A: GO SUB 33
24 FOR J=0 TO 7
25 LET A=PEEK (I+J)
26 POKE 23775,A
27 RANDOMIZE USR 23765
28 PRINT " "; GO SUB 33
29 NEXT J: PRINT " ";
30 LET A=PEEK 23776
31 GO SUB 33: PRINT
32 NEXT I: STOP
33 LET A=""
34 LET A1=INT (A/16)
35 LET A2=A-16*A1

```

```

36 LET A*=CHR* (48+A1+((A1>9)*
7))
37 LET A*=A*+CHR* (48+A2+((A2>
9)*7))
38 PRINT A*
39 RETURN
40 IF LEN A*<4 THEN LET A*=""0
"+A*": GO TO 40
41 IF LEN A*>4 THEN LET A*=A*
(2 TO ): GO TO 41
42 LET I=1: GO SUB 48
43 GO SUB 47
44 LET A=256*D: LET I=3
45 GO SUB 48: LET A=A+D
46 GO SUB 47: RETURN
47 IF D<0 OR D>255 THEN LET
A=-1: RETURN
48 LET D=((CODE a*(i)-(48+(CO
DE a*(i)>57)*7)))*16
49 LET D=D+((CODE a*(i+1)-(48
+(CODE a*(i+1)>57)*7)))
50 RETURN

```

PRIZ –adresa obsahuje příznak, který určuje přesné místo zápisu, resp. čtení, které bude použito (viz obr. 1).

Při stálém zapisování resp. čtení si program obsahy ADR a PRIZ mění sám. Když použijete konstantní obsah SLOVO, stačí vkládat délku a použitou tabulku. Při zapisování, resp. čtení z prostředka (třídění, vyhledávání, přerušené vkládání) můžete využít algoritmus pro výpočet adresy a příznaku. Obecně vypočítáme K-tou položku I-té věty (viz obr. 2).

Hlavní výhody programu:

- maximální využití paměti,
- šetří 38% paměti,
- pro zpracování je přístupno větší množství dat.

Nevýhody:

- pomalejší proces čtení, zápisu a vyhledávání (pomalejší neznamená pomalý, časové ztráty jsou vcelku zanedbatelné),
- nemožnost použití číslic a písmen v jednom údaji.

## MINIDATA

Martin Preisler, Rudé armády 684, 293 01 Mladá Boleslav

Při vytváření databází jsem začal přemýšlet jak do paměti SPECTRA vložit více dat, aniž bych rozšiřoval paměť. Nejdříve jsem zkrátil program, ale moc místa jsem tím neušetřil. Pak jsem si uvědomil, že délka jednoho ASCII znaku je 8 bitů, přičemž všechny nejsou využity (stačí si prohlédnout tabulku ASCII).

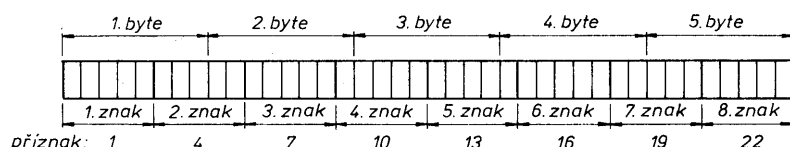
Proto jsem se rozhodl vytvořit nový znakový soubor – ve zkrácené formě. Nazval jsem jej MDAT (minidata). Abych ušetřil co největší prostor paměti, zvolil jsem jako délku jednoho znaku 5 bitů. To umožňuje ale pouze 32 znaků. Rozdělil jsem tedy údaje v databankách na nenumernické (skládají se z písmen A–Z, čárky, tečky, mezery, dvojtečky, pomlčky, apostrofu) a číselné. To vyžaduje dvojnásobek znaků.

Třicet dva číslic neexistuje, proto jsem číselné znaky doplnil matematickými operátory (krát, plus, minus, lomeno, šipka, levá a pravá závorka, desetinná tečka), některými funkcemi vkládanými v rozšířeném módu (COS, INT, ABS, SIN, SQR, LN, EXP, PI, ASN, ACS) a písmeny (A, B, C, E). Což umožní uchovávat nejrůznější vzorce pro proměnné A, B, C (E označuje exponent) a při použití funkce VAL získat přímo hodnoty.

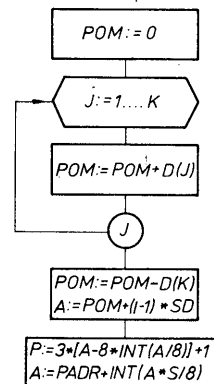
Pak jsem musel vymyslet program, který by normální ASCII znaky převáděl do paměti jako MDAT a naopak. Aby byl program rychlý, vytvořil jsem ho ve strojovém kódu mikroprocesoru Z80. Vycházel jsem z obr. 1. Takto uložených 8 znaků by se stále opakovalo. Proto jsem rozdělil program na 16 částí (8 pro zápis do paměti a 8 pro čtení) podle příznaku. Program jsem se snažil maximálně zjednodušit. Znaky, které nejsou v dané

tabulce znaků, program přepíše na „-“ (minus). Významy některých symbolů:

ZAPIS –adresa začátku programu pro zápis dat;  
 PRECTI –adresa začátku rutiny pro čtení dat;  
 TAB –adresa obsahuje adresu používané tabulky nově vytvořených znaků. V mém případě PISMO (nenumernické znaky) = adresa TAB+357 a CISLA (numerické) = adresa TAB+421. Tabulku si můžete vytvořit svou vlastní (i více), ale zachovávejte pořadí – znak, kód, posledním znakem musí být minus;  
 SLOVO –adresa obsahuje adresu, z které se při zápisu bude číst ASCII řetězec a na kterou se při čtení bude zapisovat ASCII řetězec;  
 DELKA –adresa obsahuje délku řetězce, z toho vyplývá omezení pro délku na 255 znaků;  
 ADR –adresa obsahuje adresu, na kterou se bude zapisovat, nebo ze které se bude číst;



Obr. 1. Princip ukládání znaků (919–1)



Obr. 2. Algoritmus pro výpočet adresy a příznaku (919–2)

(D(J) – délka J-tého údaje, PADR – první adresa datového souboru, P – příznak, A – adresa pro zápis-čtení znaku).

Výpis 1. Zdrojový text programu (919-V1)

919-V1

TAB	64000
SLOVO	TAB+2
ADR	TAB+4
DELKA	TAB+6
PRIZ	TAB+7
ORG	TAB+8

ZAPIS	CALL	ZAC
AI	PUSH	BC
	LD	C,(HL)
	PUSH	HL
	LD	HL,(TAB)
	CALL	NAJDI
	INC	HL
	LD	C,(HL)
	LD	HL,(ADR)
	LD	A,(HL)

A2	LD	IX,A2+1	Z6	RL	B	ZAC	DEC	HL	DEFB	'B',1
	ADD	IX,DE		ADD	A,C		LD	A,(HL)	DEFB	'C',2
	JP	(IX)		LD	(HL),A		POP	HL	DEFB	'D',3
	JP	Z1		INC	HL		LD	(HL),A	DEFB	'E',4
	JP	Z2		LD	A,(HL)		INC	HL	DEFB	'F',5
	JP	Z3		AND	1111110B		POP	BC	DEFB	'G',6
	JP	Z4		ADD	A,B		DJNZ	B1	DEFB	'H',7
	JP	Z5		JP	A3		LD	HL,PRIZ	DEFB	'I',8
JP	Z6	AND	11000001B	LD	(HL),E	DEFB	'J',9			
JP	Z7	RLC	C	RET		DEFB	'K',10			
JP	Z8	ADD	A,C			DEFB	'L',11			
A3	LD	(ADR),HL	Z7	JP	A4		LD	D,00H	DEFB	'M',12
A4	LD	(HL),A		AND	00111111B		LD	A,(DELKA)	DEFB	'N',13
	INC	E		LD	B,A		LD	B,A	DEFB	'O',14
	INC	E		LD	A,00H		LD	HL,PRIZ	DEFB	'P',15
	INC	E		RR	C		LD	E,(HL)	DEFB	'Q',16
A5	POP	HL		RRA			LD	HL,(SLOVO)	DEFB	'R',17
	INC	HL		RRC	C		RET		DEFB	'S',18
	POP	BC		RRA					DEFB	'T',19
	DJNZ	A1	ADD	A,B	NAJDI	LD	A,(HL)	DEFB	'U',20	
	LD	HL,PRIZ	LD	(HL),A		CP	45	DEFB	'V',21	
	LD	(HL),E	LD	A,C		RET	Z	DEFB	'W',22	
	RET		AND	00000111B		CP	C	DEFB	'X',23	
Z1	AND	11100000B	LD	B,A		RET	Z	DEFB	'Y',24	
	ADD	A,C	INC	HL		INC	HL	DEFB	'Z',25	
	JP	A4	LD	A,(HL)		INC	HL	DEFB	' ',26	
Z2	AND	00011111B	AND	11111000B		JP	NAJDI	DEFB	' ':27	
	LD	B,A	ADD	A,B	C2	INC	HL	DEFB	' ':28	
	LD	A,C	JP	A3		LD	B,(HL)	DEFB	' ',29	
	LD	C,00H	AND	00000111B		RR	B	DEFB	'"',30	
	RLCA		RLC	C		RRA		DEFB	'-',31	
	RLA		RLC	C	C52	RR	B			
	RLA		RLC	C		RRA				
	RLA		ADD	A,C		RRA		CISLO	DEFB 0,0	
	RL	C	LD	(HL),A		RRA			DEFB 1,1	
	RLCA		INC	HL		RRA			DEFB 2,2	
	RL	C	LD	(ADR),HL		JP	B2		DEFB 3,3	
	AND	11100000B	LD	E,01H	C3	RRA			DEFB 4,4	
	ADD	A,B	JP	A5	C6	RRA			DEFB 5,5	
	LD	(HL),A				JP	B3		DEFB 6,6	
	INC	HL	PRECTI	CALL	C4	RLCA			DEFB 7,7	
	LD	A,(HL)	B1	PUSH		INC	HL		DEFB 8,8	
	AND	11111100B		PUSH		LD	A,(HL)		DEFB 9,9	
	ADD	A,C		LD		RLA			DEFB '+',10	
	JP	A3		LD		JP	B2		DEFB ' ',11	
Z3	AND	10000011B		LD		INC	HL		DEFB '+',12	
	RLC	C		ADD		LD	B,(HL)		DEFB ' ',13	
	RLC	C	XX	JP		JP	C52		DEFB ' ',14	
	ADD	A,C		JP		INC	HL		DEFB '(',15	
	JP	A4		JP		LD	B,(HL)		DEFB ')',16	
Z4	RLCA			JP		RLCA			DEFB 'A',17	
	RR	C		JP		RL	B		DEFB 'B',18	
	RRA			JP		RLCA			DEFB 'C',19	
	LD	(HL),A		JP		RL	B		DEFB 'E',20	
	INC	HL		JP		LD	A,B		DEFB 179,21;COS	
	LD	A,(HL)		JP		JP	B2		DEFB 186,22;INT	
	AND	11110000B	B2	LD		RRA			DEFB 189,23;ABS	
	ADD	A,C	B3	INC		RRA			DEFB 178,24;SIN	
	JP	A3		INC		RRA			DEFB 187,25;SQR	
Z5	LD	B,00H		INC		INC	HL		DEFB 184,26;LN	
	AND	00001111B	B4	AND		LD	(ADR),HL		DEFB 185,27;EXP	
	RLC	C		LD		LD	E,1		DEFB 167,28;PI	
	RLC	C		LD		JP	B4		DEFB 181,29;ASN	
	RLC	C		INC					DEFB 182,30;ACS	
	SLA	C		CALL		PISMO	DEFB 'A',0		DEFB ' ',31	
				NAJDI						

# GRAFIKA V PASCALU

Pavel Kríž, Paláskova 1107/2, 180 00 Praha 8

Grafika je souhrn podprogramů tvořících grafický systém. S jejich pomocí můžete snadno vytvářet různé obrázky, schémata, grafy apod. Podprogramy umožňují transformovat souřadný systém, kreslit čáry a jednoduché obrazce, měnit barvy, vyplňovat uzavřené obrazce. Kreslit můžete na celou plochu obrazovky o velikosti 256×192 bodů.

Systém je vytvořen jako knihovna podprogramů přizpůsobená pro Hisoft Pascal 4T. Tuto knihovnu můžete dále podle své potřeby rozšiřovat.

## Přehled podprogramů

Podle činnosti lze podprogramy rozdělit do pěti částí:

### 1) Organizační podprogramy:

<b>GINIT</b>	nastaví implicitní hodnoty,
<b>WINDOW</b>	definuje velikost okénka pro kreslení,
<b>CLS</b>	vymaže obrazovku,
<b>CLW</b>	vymaže okénko,
<b>LTYPE</b>	definuje druh čar,
<b>DTYPE</b>	definuje způsob zobrazení bodů a čar,
<b>PATTERN</b>	definuje vzorek pro vyplňování a jeho rotaci,
<b>AT</b>	nastaví polohu tisku,
<b>INK,PAPER</b>	nastaví barvy pro kreslení,
<b>BRIGHT,FLASH</b>	nastaví barvy pro kreslení,
<b>BORDER</b>	změní barvu okraje obrazovky.

### 2) Transformační podprogramy:

<b>SCALE</b>	nastaví měřítko os,
<b>COORD</b>	nastaví počátek souřadného systému,
<b>TURN</b>	natočí souřadný systém,
<b>SYMMET</b>	nastaví symetrii,
<b>SHIFT</b>	provede posunutí od počátku COORD.

### 3) Kreslicí podprogramy:

<b>CURSOR</b>	nastaví kurzor,
<b>PLOT</b>	nakreslí bod,
<b>DRAW,LINETO,</b>	
<b>LINE</b>	nakreslí přímku,
<b>TRIA</b>	nakreslí trojúhelník,
<b>TETRA</b>	nakreslí čtyřúhelník,
<b>BOX</b>	nakreslí obdélník rovnoběžný s osami x a y,
<b>BOW</b>	nakreslí oblouk,
<b>CIRCLE</b>	nakreslí kružnici,
<b>FILL</b>	vyplní uzavřený obrazec,
<b>PAINT</b>	vyplní uzavřený obrazec zadaným vzorkem.

### 4) Funkce:

<b>POSIT</b>	poloha kurzoru,
<b>POINT</b>	informace o bodu.

### 5) Interně používané procedury a funkce:

V této části jsou podprogramy přizpůsobené a pomocné podprogramy používané v částech 1 až 4. Pro úplnost uvádím i jejich seznam:

**CURSOR0, DRAW0, FILLO, INVERSE, OVER, PAINT0, POINT0, RANGE, ROT, TRANS, TRCONST.**

Systém používá globální proměnné:

**GDTYPE, GLTYPE, GPAT, GROT, XMAN, XMIN, XOS, XSYM, YMAX, YMIN, YOS,**

**YSYM: INTEGER;**  
**GALFA, GDETER, XCO, XSC, XSH, XTR1, XTR2, XTR3, YCO, YSC, YSH, YTR1, YTR2, YTR3: REAL.**

## Popis podprogramů

### 1) Organizační podprogramy:

**GINIT**  
 Tento podprogram musíte vyvolat před použitím ostatních podprogramů. GINIT provede nastavení implicitních hodnot:

**WINDOW (0,0,255,191,0,0);** celá obrazovka.  
**SCALE (1,1);** měřítko 1:1 pro x i y.  
**COORD (0,0);** počátek zůstane v levém dolním rohu.  
**SHIFT (0,0);** osy jsou totožné s okrajem obrazovky.  
**SYMMET (0,0); TYRN (0);** plná čára.  
**DTYPE (1); LTYPE (1);** jednoduchý vzorek.  
**PATTERN (1,1);**

**WINDOW (XMIN,YMIN,XMAX,YMAX, X0,Y0: INTEGER)**

Definuje polohu okénka na obrazovce. Střed souřadnic se nastaví ve vzdálenosti  $|X0,Y0|$  od levého dolního rohu okénka. Parametry jsou nezávislé na nastaveném měřítku.

Parametry musíte zadat tak, aby platilo

$0 \leq XMIN \leq XMAX \leq 255$      $0 \leq X0 \leq XMAX - XMIN$   
 $0 \leq YMIN \leq YMAX \leq 255$      $0 \leq Y0 \leq YMAX - YMIN$

### CLS

Vymaže obrazovku, přenesse dočasné barvy do trvalých a vyplní jimi obrazovku. Nastaví tisk do levého horního rohu, kurzor pro kreslení nastaví do bodu  $[0,0]$ .

### CLW (FRAME: BOOLEAN)

Vymaže okénko definované příkazem WINDOW. Parametrem FRAME si zvolíte, má-li se okénko ohraničit rámečkem (TRUE).

### LTYPE (LT: INTEGER)

Definuje druh čar:

1 ..... plná čára,  
 2 ..... čárkovaná čára,  
 3 ..... tečkovaná čára.

### DTYPE (DT: INTEGER)

Definuje způsob zobrazení bodů a čar:  
 0 ..... vymazávání (INVERSE 1),  
 1 ..... plná čára,  
 2 ..... změna bodů (OVER 1).

### PATTERN (PAT,RP: INTEGER)

Těmito parametry se řídí způsob vyplňování uzavřeného obrazce podprogramem PAINT.

PAT je osmibitový vzorek, který je v binárním tvaru „naskládán“ do vodorovné linky. RP ovlivňuje rotaci vzoru při přechodu na vyšší linku. Vzorek se posune o RP bitů vpravo (pro  $RP < 0$ ) nebo o  $|RP|$  bitů vlevo.

Parametry musí být v rozmezí

$0 \leq PAT \leq 255$

$-7 \leq RP \leq 7$ .

Příklady vzorků: BIN 01010101 = 85,  
 BIN 00010001 = 17.

### AT (LIN,COL: INTEGER)

Nastaví tiskovou pozici na řádek LIN a sloupec COL.

### INK (I: INTEGER)

### PAPER (P: INTEGER)

### BRIGHT (B: INTEGER)

### FLASH (F: INTEGER)

Těmito podprogramy se nastaví dočasné aktivní barvy. Dovolené parametry jsou 0 až 9 pro INK a PAPER, 0,1,8 pro BRIGHT a FLASH. Význam parametrů viz manuál ZX Spectra.

### BORDER (B: INTEGER)

Tímto podprogramem změníte barvu okraje obrazovky. Dovolené hodnoty jsou 0 až 7.

### 2) Transformační podprogramy:

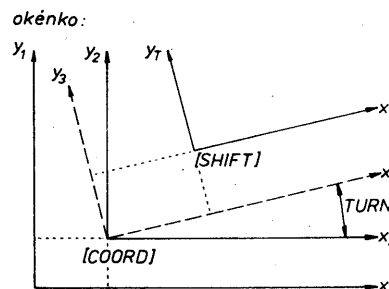
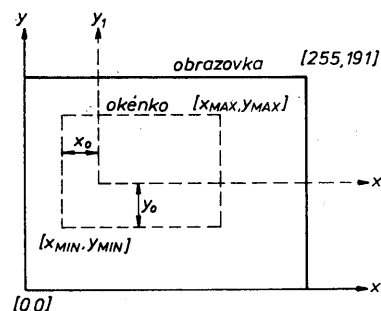
### SCALE (SX,SY: REAL)

Nastaví měřítko SX:1 pro osu x a SY:1 pro osu y. Nepřípustným parametrem je 0. Měřítko ovlivňuje dále uvedenou transformaci.

### COORD (X0,Y0: REAL)

Nastaví počátek souřadného systému.  $X0,Y0$  je posunutí od středu zadaného podprogramem WINDOW.

K bodu COORD se vztahují dále uvedená transformace.



Obr. 1. Znárodnění transformací (926-1)

## TURN (ALFA: REAL)

Souřadný systém se otočí okolo bodu COORD o ALFA stupňů (proti směru hodinových ručiček).

## SYMMET (SX,SY: INTEGER)

Nastaví se symetrie:

- (1,0) ..... podle osy x,
- (0,1) ..... podle osy y,
- (1,1) ..... podle počátku,
- (0,0) ..... zrušení symetrie.

## SHIFT (SX,SY: REAL)

V natočeném systému souřadnic se bod |0,0| posune ve vzdálenosti |SX,SY| od bodu COORD.

Uvedené transformační podprogramy provádějí pouze nastavování potřebných hodnot. Při opakovaném volání platí vždy poslední zadaná hodnota, na níž předchozí nemá žádný vliv. Na pořadí volání transformačních podprogramů nezáleží.

Transformace se zruší vyvoláním podprogramu s nulovými hodnotami (pro měřítko s hodnotami (1,1)).

Znárodnění transformací je na obr. 1.

### 3) Kreslicí podprogramy:

## CURSOR (X,Y: REAL)

Přenes kurzor do bodu |X,Y|.

V dále uvedených podprogramech se kurzor nastaví do posledního nakresleného bodu, u obrazců (TRIA, TETRA, BOX) zůstane v prvním zadaném bodě, po vyplnění (FILL, PAINT) se nastaví do zadaného vnitřního bodu.

## PLOT (X,Y: REAL)

Nakreslí bod |X,Y|.

## DRAW (DX,DY: REAL)

Nakreslí přímkou od kurzoru do bodu |Xkurzor+DX, Ykurzor+DY|.

## LINE (X1,Y1,X2,Y2: REAL)

Nakreslí přímkou od kurzoru do bodu |X2, Y2|.

## LINE (X1,Y1,X2,Y2: REAL)

Nakreslí přímkou z bodu |X1,Y1| do bodu |X2,Y2|.

## TRIA (X1,Y1,X2,Y2,X3,Y3,X4,Y4: REAL)

Nakreslí trojúhelník zadaný třemi body.

## TETRA (X1,Y1,X2,Y2,X3,Y3,X4,Y4: REAL)

Nakreslí čtyřúhelník zadaný čtyřmi body.

## BOX (X1mY1,X2,Y2: REAL)

Nakreslí obdélník rovnoběžný s osami x a y, přičemž body |X1,Y1|, |X2,Y2| jsou souřadnice úhlopříčky.

## BOW (X,Y,R,ALFA,BETA: REAL)

Nakreslí oblouk se středem v bodě |X,Y| a poloměrem R. Začátek oblouku svírá s poloosou X+ úhel ALFA. BETA je vnitřní úhel oblouku. (Hodnoty ALFA, BETA jsou ve stupních.)

## CIRCLE (X,Y,R: REAL)

Nakreslí kružnici se středem v |X,Y| a poloměrem R. Pokud budou měřítka pro osu x a y rozdílná, bude výsledkem ve skutečnosti elipsa.

## FILL (X,Y: REAL)

Vyplní uzavřený obrazec barvou inkoustu. Bod |X,Y| je uvnitř tohoto obrazce.

## PAINT (X,Y: REAL)

Vyplní uzavřený obrazec vzorkem zadaným podprogramem PATTERN. Bod |X,Y| leží opět uvnitř tohoto obrazce.

**Poznámka:** Při větších a složitějších obrazcích jsou podprogramy FILL a PAINT

náročnější na volnou paměť. Tyto požadavky ovlivníte kladně tím, že vnitřní bod zadáte uprostřed vyplňovaného obrazce.

Parametry všech kreslicích podprogramů se zadávají v transformovaných souřadnicích. Pokud bude bod ležet mimo zadané okénko, vypíše se zpráva Out of window X=... Y=... XT=... YT=..., kde XT a YT jsou hodnoty transformovaných souřadnic, X a Y jsou hodnoty přepočítané do skutečných souřadnic. Po vypisání této chybové zprávy se program zastaví se zprávou Halt at PC=...

### 4) Funkce:

## POSIT (VAR, PX, PY: REAL)

Ve skutečnosti nejde o funkci, ale o proceduru, která vrací pozici kurzoru v transformovaných souřadnicích.

## POINT (X,Y: REAL): BOOLEAN

Tato funkce vrací informaci o bodu:

TRUE .... bod je vyplněn  
barvou inkoustu,  
FALSE ... bod není vyplněn.

## Použití grafiky

Tato verze systému grafiky je přizpůsobena pro HiSoft Pascal. Zdrojový text systému s názvem grafR.P je dlouhý 7485 bajtů. Vytvořený cílový kód je dlouhý 8715 bajtů.

Po nahrání HiSoft Pascalu musíte zadat tabulku symbolů (Table size). Celý grafický systém spotřebuje pro tabulku 2600 bajtů. Bohužel většina verzí HP dovoluje použít pro tabulku symbolů max. 2200 bajtů. Potřebnou hodnotu 2600 i více lze zadat u verze HP 113 (P. Adámek).

Uvedené hodnoty naznačují, že v paměti nezbyvá již mnoho místa (HP zabírá přes 20 kB). Zdrojový text grafiky proto nelze kompilovat celý. Z programu můžete vymazat části, které nebudete potřebovat a pak zkompilovat.

Aby se paměť nemusela zbytečně zatěžovat uchováváním zdrojového textu, je výhodné použít kompilaci z pásky. To umožňuje kompilovat celý grafický systém. Proto jsem vytvořil „knihovnu verzí“.

Systém grafika je zde rozdělen do 5 částí:

1. **grafR0.W** – velikost cílového kódu je 3506 bajtů.

Základní část, obsahující podprogramy přizpůsobení, organizační podprogramy (GINIT, CLS, LTYPE, DTYPE, PATTERN, AT), kreslicí podprogramy (CURSOR, PLOT, DRAW, LINE, LINE TO);

2. **grafR1.W** – 811 bajtů.

Obsahuje organizační podprogramy (WINDOW, CLW), transformační podprogramy (SCALE, COORD, TURN, SYMMET, SHIFT);

3. **grafR2.W** – 2031 bajtů.

Obsahuje kreslicí podprogramy (TRIA, TETRA, BOX, BOW, CIRCLE);

4. **grafR3.W** – 2217 bajtů.

Obsahuje kreslicí podprogramy (FILL, PAINT);

5. **colour.W** – 150 bajtů.

Obsahuje podprogramy barev.

**Poznámka:** označení .W v názvu znamená, že nejde o standardní formát zdrojového textu, ale o knihovnu verzí nahranou editorem povelu W.

Při použití GRAFIKY zkompilujte část grafR0. a k ní pak přidejte další potřebnou

část. Mezi deklaraci proměnných a deklaraci procedur a funkcí ve svém programu vložte řádky:

```
{ SF grafR0.W }  
{ SF grafRx.W },
```

kde x je číslo části.

Pokud nemáte ve svém programu žádnou proměnnou, dejte před tento řádek klíčové slovo VAR.

Při spuštění kompilace editorovým povelu C nebo T se u řádku s direktivou F kompilace zastaví a je připravena na vkládání z pásky.

## Verze Integer

Pro jednodušší kreslení jsem vytvořil verzi INTEGER, která je kratší a vytváří i menší cílový kód. Zdrojový kód s názvem grafI.P je dlouhý 5744 bajtů, velikost cílového kódu je 5376 bajtů. Pro tabulku symbolů stačí 2200 bajtů. Vyhovuje tedy všem verzím HiSoft Pascalu.

Odlišnosti od verze REAL:

I. Kreslit můžete na obrazovku o velikosti 256×176 bodů.

II. Všechny uvedené podprogramy, které měly parametry typu REAL, mají zde parametry typu INTEGER.

III. Verze Integer neobsahuje tyto podprogramy:  
LTYPE – definice typu čar,  
SCALE – nastavení měřítka,  
TURN – natočení souřadného systému,  
BOW – kreslení oblouku.

IV. Knihovna má následující části:

1. **grafI0.W** – velikost cílového kódu je 1894 bajtů.

Základní část, obsahující podprogramy přizpůsobení, organizační podprogramy (GINIT, CLS, DTYPE, PATTERN AT), kreslicí podprogramy (CURSOR, PLOT, DRAW, LINE, LINE TO, CIRCLE);

2. **grafI1.W** – 584 bajtů.

Obsahuje organizační podprogramy (WINDOW, CLW), transformační podprogramy (COORD, SYMMET, SHIFT);

3. **grafI2.W** – 604 bajtů.

Obsahuje kreslicí podprogramy (TRIA, TETRA, BOX);

4. **grafI3.W** – 2144 bajtů.

Obsahuje kreslicí podprogramy (FILL, PAINT);

5. **colour.W** – 150 bajtů.

Obsahuje podprogramy barev (stejně jako u verze REAL).

Všechny programy, napsané pro verzi Integer, budou pracovat stejně i s verzí Real.

## Ukázky programů

### 1. Schody (Výpis 3.)

Tento program ukazuje použití okénka, změnu měřítka, transformace souřadnic a vyplňování.

### 2. Průběh funkce (Výpis 4.)

Nakreslí se průběh jedné periody funkce Y=SIN(X). Ukazuje výhody změny měřítka.

### 3. Pneumatika (Výpis 5.)

Program nakreslí zjednodušené pneumatiku a vybarví ji. Je použita verze INTEGER grafického systému.

Výpis 1. Grafický systém, verze REAL  
(926-VI)

```

1  {
2  {   GRAFICKÝ SYSTÉM   }
3  {   P.KRIZ (c) 1988   }
4  {   verze REAL       }
5  {
6  { $L- }
7  XMIN,YMIN,XMAX,YMAX,XOS,YOS,XYSYM,YSYM,
8  GPAT,GROT,GDTYPE,GLTYPE:INTEGER;
9  XCO,YCO,XSC,YSC,XSH,YSH,GALFA,GDETER,
10 XTR1,XTR2,XTR3,YTR1,YTR2,YTR3:REAL;
11 { $O- ,C-,S-,I- }
12 PROCEDURE CURSOR(X,Y:INTEGER);
13 BEGIN
14   INLINE(221,102,2,221,110,4,34,125,92)
15 END;
16 PROCEDURE PLOT(X,Y:INTEGER);
17 BEGIN
18   INLINE(253,33,58,92,221,70,2,221,78,4,237);
19   INLINE(67,125,92,62,191,205,172,34,205,236,34)
20 END;
21 PROCEDURE DRAW(DX,DY:INTEGER);
22 BEGIN
23   INLINE(253,33,58,92,17,1,1,221,70,2);
24   INLINE(221,124,3,167,40,4,87,175,144,71);
25   INLINE(221,78,4,221,126,5,167,40,4,95);
26   INLINE(175,145,79,121,184,48,4,105,213,175);
27   INLINE(95,24,8,177,40,51,104,65,213,22);
28   INLINE(0,96,120,31,133,56,3,188,56,7);
29   INLINE(148,79,217,193,197,24,4,79,213,217);
30   INLINE(193,42,125,92,121,133,79,120,132,71);
31   INLINE(237,67,125,92,62,191,205,172,34,205);
32   INLINE(236,34,217,121,16,214,209)
33 END;
34 FUNCTION POINT(X,Y:INTEGER):BOOLEAN;
35 BEGIN
36   INLINE(221,70,2,221,78,4,62,191,205,172,34);
37   INLINE(71,4,126,7,16,253,230,1,221,119,6)
38 END;
39 PROCEDURE POSIT(VAR PX,PY:INTEGER);
40 BEGIN
41   INLINE(237,91,125,92,175,221,110,4);
42   INLINE(221,102,5,115,35,119,221,110,2);
43   INLINE(221,102,3,114,35,119)
44 END;
45 PROCEDURE OVER(O:INTEGER);
46 BEGIN
47   INLINE(62,2,221,86,2,205,29,34)
48 END;
49 PROCEDURE INVERSE(I:INTEGER);
50 BEGIN
51   INLINE(62,1,221,86,2,205,29,34)
52 END;
53 PROCEDURE AT(LIN,COL:INTEGER);
54 BEGIN
55   INLINE(221,70,2,221,78,4,205,10,32)
56 END;
57 PROCEDURE TRANS(X,Y:REAL;VAR PX,PY:INTEGER);
58 BEGIN
59   PX:=ROUND(XTR1+XTR2*X+XTR3*Y);
60   PY:=ROUND(YTR1+YTR2*X+YTR3*Y);
61   IF (PX<XMIN) OR (PX>XMAX) OR (PY<YMIN) OR (PY>YMAX) THEN
62     BEGIN
63       WRITE(CHR(13),'Out of window: X=',PX,' Y=',PY,' XT=',X,
64       YT=',Y);HALT
65     END
66   END;
67 PROCEDURE POSIT(VAR PX,PY:REAL);
68 VAR X,Y:INTEGER;
69 BEGIN
70   POSIT(X,Y);
71   PX:=((X-XTR1)*YTR3-(Y-YTR1)*XTR3)/GDETER;
72   PY:=((Y-YTR1)*XTR2-(X-XTR1)*YTR2)/GDETER
73 END;
74 PROCEDURE TRCONST;
75 BEGIN
76   XTR2:=YSYM*XSC*COS(GALFA);
77   YTR2:=YSYM*XSC*SIN(GALFA);
78   XTR3:=-XSYM*YSC*SIN(GALFA);
79   YTR3:=XSYM*YSC*COS(GALFA);
80   XTR1:=XOS+XSC*XCO+XTR2*XSH+XTR3*YSH;
81   YTR1:=YOS+YSC*YCO+YTR2*XSH+YTR3*YSH;
82   GDETER:=XTR2*YTR3-XTR3*YTR2
83 END;
84 PROCEDURE LTYPE(LT:INTEGER);
85 BEGIN
86   GLTYPE:=LT
87 END;
88 PROCEDURE DTYPE(DT:INTEGER);
89 BEGIN
90   GDTYPE:=DT;
91   OVER(0);INVERSE(0);
92   IF DT=0 THEN INVERSE(1);
93   IF DT=2 THEN OVER(1)
94 END;
95 PROCEDURE PATTERN(PAT,RP:INTEGER);
96 BEGIN
97   GPAT:=PAT;GROT:=RP
98 END;
99 FUNCTION POINT(X,Y:REAL):BOOLEAN;
100 VAR PX,PY:INTEGER;
101 BEGIN
102   TRANS(X,Y,PX,PY);
103   POINT:=POINT(PX,PY)
104 END;
105 PROCEDURE CURSOR(X,Y:REAL);
106 VAR PX,PY:INTEGER;
107 BEGIN
108   TRANS(X,Y,PX,PY);
109   CURSOR(PX,PY)
110 END;
111 PROCEDURE PLOT(X,Y:REAL);
112 VAR PX,PY:INTEGER;
113 BEGIN
114   TRANS(X,Y,PX,PY);
115   PLOT(PX,PY)
116 END;
117 PROCEDURE DRAW(DX,DY:REAL);
118 VAR A,B,N,X,Y,I:INTEGER;
119   XR,YR,PDX,PDY,L:REAL;
120 POSIT(XR,YR);

```

```

121 TRANS(XR+DX,YR+DY,X,Y);
122 PDX:=XTR2*DX+XTR3*DY;
123 PDY:=YTR2*DX+YTR3*DY;
124 IF GLTYPE=1 THEN DRAW(ROUND(PDX),ROUND(PDY))
125 ELSE BEGIN
126   IF GLTYPE=2 THEN BEGIN A:=2;B:=4 END
127   ELSE BEGIN A:=4;B:=0 END;
128   L:=SQRT(SQR(PDX)+SQR(PDY));
129   N:=TRUNC((L+B)/(A+B));
130   POSIT(X,Y);
131   FOR I:=0 TO N-1 DO BEGIN
132     PLOT(X+ROUND(PDX*I/N),Y+ROUND(PDY*I/N));
133     IF GLTYPE=2 THEN
134       DRAW(ROUND(PDX*A/L),ROUND(PDY*A/L))
135   END;
136   PLOT(X+ROUND(PDX),Y+ROUND(PDY))
137 END;
138 END;
139 PROCEDURE LINETO(X2,Y2:REAL);
140 VAR X1,Y1:REAL;
141 BEGIN
142   POSIT(X1,Y1);DRAW(X2-X1,Y2-Y1)
143 END;
144 PROCEDURE LINE(X1,Y1,X2,Y2:REAL);
145 BEGIN
146   PLOT(X1,Y1);DRAW(X2-X1,Y2-Y1)
147 END;
148 PROCEDURE CLS;
149 BEGIN
150   PAGE:AT(0,0);CURSOR(0,0);
151   INLINE(205,173,28,58,141,92,33,0,88);
152   INLINE(17,1,88,1,255,2,119,237,176)
153 END;
154 PROCEDURE GINIT;
155 BEGIN
156   XSC:=1;YSC:=1;XCO:=0;YCO:=0;GALFA:=0;
157   XSH:=0;YSH:=0;XYSYM:=1;YSYM:=1;XOS:=0;YOS:=0;
158   XMIN:=0;YMIN:=0;XMAX:=255;YMAX:=191;
159   TRCONST;
160   CURSOR(0,0);
161   DTYPE(1);LTYPE(1);
162   PATTERN(1,1)
163 END;
164 PROCEDURE WINDOW(X1,Y1,X2,Y2,X0,Y0:INTEGER);
165 BEGIN
166   XMIN:=X1;YMIN:=Y1;XMAX:=X2;YMAX:=Y2;XOS:=XMIN
167   +X0;YOS:=YMI N+Y0;
168   TRCONST
169 END;
170 PROCEDURE COORD(X0,Y0:REAL);
171 BEGIN
172   XCO:=X0;YCO:=Y0;TRCONST
173 END;
174 PROCEDURE SCALE(SX,SY:REAL);
175 BEGIN
176   XSC:=SX;YSC:=SY;TRCONST
177 END;
178 PROCEDURE SHIFT(SX,SY:REAL);
179 BEGIN
180   XSH:=SX;YSH:=SY;TRCONST
181 END;
182 PROCEDURE SYMMET(SX,SY:INTEGER);
183 BEGIN
184   IF SX=0 THEN XSYM:=1 ELSE XSYM:=-1;
185   IF SY=0 THEN YSYM:=1 ELSE YSYM:=-1;
186   TRCONST
187 END;
188 PROCEDURE TURN(ALFA:REAL);
189 BEGIN
190   GALFA:=ALFA*3.1416/180;TRCONST
191 END;
192 PROCEDURE CLW(FRAME:BOOLEAN);
193 VAR Y,T:INTEGER;
194 BEGIN
195   T:=GDTYPE;DTYPE(0);
196   FOR Y:=YMIN TO YMAX DO BEGIN
197     PLOT(XMIN,Y);DRAW(XMAX-XMIN,0)
198   END;
199   DTYPE(T);
200   IF FRAME THEN BEGIN
201     CURSOR(XMIN,YMIN);
202     DRAW(0,YMAX-YMIN);DRAW(XMAX-XMIN,0);
203     DRAW(0,YMIN-YMAX);DRAW(XMIN-XMAX,0)
204   END;
205   CURSOR(0,0)
206 END;
207 PROCEDURE TRIA(X1,Y1,X2,Y2,X3,Y3:REAL);
208 BEGIN
209   CURSOR(X1,Y1);
210   DRAW(X2-X1,Y2-Y1);
211   DRAW(X3-X2,Y3-Y2);
212   DRAW(X1-X3,Y1-Y3)
213 END;
214 PROCEDURE TETRA(X1,Y1,X2,Y2,X3,Y3,X4,Y4:REAL);
215 BEGIN
216   CURSOR(X1,Y1);
217   DRAW(X2-X1,Y2-Y1);
218   DRAW(X3-X2,Y3-Y2);
219   DRAW(X4-X3,Y4-Y3);
220   DRAW(X1-X4,Y1-Y4)
221 END;
222 PROCEDURE BOX(X1,Y1,X2,Y2:REAL);
223 BEGIN
224   CURSOR(X1,Y1);
225   DRAW(0,Y2-Y1);
226   DRAW(X2-X1,0);
227   DRAW(0,Y1-Y2);
228   DRAW(X1-X2,0)
229 END;
230 PROCEDURE BOW(X,Y,R,ALFA,BETA:REAL);
231 VAR U,V,RX,RY,RM:REAL;
232 BEGIN
233   ALFA:=ALFA*3.1416/180;
234   BETA:=BETA*3.1416/180;
235   CURSOR(X+R*COS(ALFA),Y+R*SIN(ALFA));
236   RX:=ABS(R*XSC);RY:=ABS(R*YSC);
237   IF RX>RY THEN RM:=RX ELSE RM:=RY;
238   U:=3.1416/36;
239   IF RM<10 THEN U:=2*U;
240   IF RM<25 THEN U:=3*U;

```

```

240 V:=U;
241 WHILE V<=BETA DO BEGIN
242   LINETO(X+R*COS(ALFA+V),Y+R*SIN(ALFA+V));
243   V:=V+U
244 END;
245 LINETO(X+R*COS(ALFA+BETA),Y+R*SIN(ALFA+BETA));
246 END;
247 PROCEDURE CIRCLE(X,Y,R:REAL);
248 BEGIN
249   ROW(X,Y,R,0,360)
250 END;
251 PROCEDURE RANGE(VAR X1,X2:INTEGER; Y:INTEGER);
252 BEGIN
253   IF X1>XMIN THEN
254     REPEAT X1:=X1-1
255     UNTIL (X1=XMIN) OR POINT0(X1,Y);
256   IF X2<XMAX THEN
257     REPEAT X2:=X2+1
258     UNTIL (X2=XMAX) OR POINT0(X2,Y)
259 END;
260 {S+}
261 PROCEDURE FILL0(VAR X:INTEGER; Y:INTEGER);
262 VAR X1,X2:INTEGER;
263 BEGIN
264   X2:=X;
265   RANGE(X,X2,Y);
266   CURSOR0(X,Y);
267   X:=X+1;
268   DRAW0(X-X,0);
269   X1:=X;
270   IF Y<YMAX THEN
271     WHILE X<X2 DO BEGIN
272       IF NOT POINT0(X,Y+1) THEN FILL0(X,Y+1);
273       X:=X+1
274     END;
275     X:=X1;
276     IF Y>YMIN THEN
277       WHILE X<X2 DO BEGIN
278         IF NOT POINT0(X,Y-1) THEN FILL0(X,Y-1);
279         X:=X+1
280       END;
281       X:=X1;
282     END;
283   {S-}
284 PROCEDURE FILL(X,Y:REAL);
285 VAR PX,PY,X1,T:INTEGER;
286 BEGIN
287   TRANS(X,Y,PX,PY);
288   IF NOT POINT0(PX,PY) THEN BEGIN
289     T:=GDTYPE;DTYPE(1);
290     FILL0(PX,PY);
291     DTYPE(T)
292   END;
293 END;
294 CURSOR0(X1,PY)
295 END;
296 {S+}
297 PROCEDURE PAINT0(X,Y,PAT:INTEGER);
298 VAR X1,X2,PAT1:INTEGER;
299 FUNCTION ROT(PAT,N:INTEGER):INTEGER;
300 VAR I:INTEGER;
301 BEGIN
302   FOR I:=1 TO N DO BEGIN
303     PAT:=PAT*2;
304     IF PAT>255 THEN PAT:=PAT-255
305   END;
306   FOR I:=1 TO N DO BEGIN
307     IF PAT MOD 2=1 THEN PAT:=PAT+256;
308     PAT:=PAT DIV 2
309   END;
310   ROT:=PAT
311 END;
312 BEGIN
313   X1:=X;X2:=X;
314   RANGE(X1,X2,Y);
315   X1:=X1+1;X2:=X2-1;
316   PAT:=ROT(PAT,(X-X1) MOD 8);
317   PAT1:=PAT;
318   FOR X:=X1 TO X2 DO BEGIN
319     PAT:=PAT*2;
320     IF PAT>255 THEN BEGIN
321       PAT:=PAT-255;
322       PLOT0(X,Y)
323     END;
324   END;
325   IF Y<YMAX THEN BEGIN
326     PAT:=ROT(PAT1,XSYM*YSYM*GROT);
327     FOR X:=X1 TO X2 DO BEGIN
328       IF (PAT>127) AND NOT POINT0(X,Y+1) THEN PAINT0(X,Y+1,PAT);
329       PAT:=PAT*2;
330       IF PAT>255 THEN PAT:=PAT-255
331     END;
332   END;
333   IF Y>YMIN THEN BEGIN
334     PAT:=ROT(PAT1,-XSYM*YSYM*GROT);
335     FOR X:=X1 TO X2 DO BEGIN
336       IF (PAT>127) AND NOT POINT0(X,Y-1) THEN PAINT0(X,Y-1,PAT);
337       PAT:=PAT*2;
338       IF PAT>255 THEN PAT:=PAT-255
339     END;
340   END;
341 END;
342 {S-}
343 PROCEDURE PAINT(X,Y:REAL);
344 VAR PX,PY,T:INTEGER;
345 BEGIN
346   TRANS(X,Y,PX,PY);
347   IF NOT POINT0(PX,PY) THEN BEGIN
348     T:=GDTYPE;DTYPE(1);
349     PAINT0(PX,PY,GPAT);
350     DTYPE(T)
351   END;
352 END;
353 CURSOR0(PX,PY)
354 END;
355 PROCEDURE INK(I:INTEGER);
356 BEGIN
357   INLINE(55,221,86,2,205,52,34)
358 END;
359 PROCEDURE PAPER(P:INTEGER);
360 BEGIN
361   INLINE(167,221,86,2,205,52,34)
362 END;
363 PROCEDURE BRIGHT(B:INTEGER);
364 BEGIN
365   INLINE(151,221,86,2,205,116,34)
366 END;

```

```

366 PROCEDURE FLASH(F:INTEGER);
367 BEGIN
368   INLINE(175,60,221,86,2,205,29,34)
369 END;
370 PROCEDURE BORDER(B:INTEGER);
371 BEGIN
372   INLINE(221,126,2,205,151,34)
373 END;
374 {L+,0+,C+,S+,I-}

```

## Výpis 2. Grafický systém, verze INTEGER (926-V2)

```

1 {
2   GRAFICKY SYSTEM
3   P.KRIZ (c) 1988
4   verze INTEGER
5 }
6 {L-}
7 XCO,YCO,XSYM,YSYM,XMIN,YMIN,XMAX,YMAX,
8 XOS,YOS,XSH,YSH,GPAT,GROT,GDTYPE: INTEGER;
9 {0-,C-,S-,I-}
10 PROCEDURE CURSOR0(X,Y:INTEGER);
11 BEGIN
12   INLINE(221,102,2,221,110,4,34,125,92)
13 END;
14 PROCEDURE PLOT0(X,Y:INTEGER);
15 BEGIN
16   INLINE(253,33,58,92,221,70,2,221,78,4,205,229,34)
17 END;
18 PROCEDURE DRAW0(DX,DY:INTEGER);
19 BEGIN
20   INLINE(253,33,58,92,17,1,1,221,70,2);
21   INLINE(221,126,3,167,40,4,87,175);
22   INLINE(144,71,221,78,4,221,126,3,167);
23   INLINE(40,4,95,175,145,79,205,188,36)
24 END;
25 PROCEDURE CIRCLE0(X,Y,R:INTEGER);
26 BEGIN
27   INLINE(221,126,6,205,40,45);
28   INLINE(221,126,4,205,40,45);
29   INLINE(221,126,2,205,40,45);
30   INLINE(205,45,35)
31 END;
32 FUNCTION POINT0(X,Y:INTEGER):BOOLEAN;
33 BEGIN
34   INLINE(221,70,2,221,78,4);
35   INLINE(205,170,34,71,4,126,7);
36   INLINE(16,253,230,1,221,119,6)
37 END;
38 PROCEDURE POSIT0(VAR PX,PY:INTEGER);
39 BEGIN
40   INLINE(237,91,125,92,175,221,110,4);
41   INLINE(221,102,5,115,35,119,221,110,2);
42   INLINE(221,102,3,114,35,119)
43 END;
44 PROCEDURE OVER(O:INTEGER);
45 BEGIN
46   INLINE(62,2,221,86,2,205,29,34)
47 END;
48 PROCEDURE INVERSE(I:INTEGER);
49 BEGIN
50   INLINE(62,1,221,86,2,205,29,34)
51 END;
52 PROCEDURE AT(LIN,COL:INTEGER);
53 BEGIN
54   INLINE(221,70,2,221,78,4,205,10,32)
55 END;
56 PROCEDURE TRANS(VAR PX,PY:INTEGER);
57 BEGIN
58   PX:=(PX+XSH)*YSYM+XCO+XOS;
59   PY:=(PY+YSH)*XSYM+YCO+YOS;
60 IF (PX<XMIN) OR (PX>XMAX) OR (PY<YMIN) OR (PY>YMAX) THEN
61 BEGIN WRITE(CHR(13),'Out of window: X=',PX,'Y=',PY);HALT
62 END;
63 FUNCTION POINT(X,Y:INTEGER):BOOLEAN;
64 BEGIN
65   TRANS(X,Y);POINT:=POINT0(X,Y)
66 END;
67 PROCEDURE POSIT(VAR PX,PY:INTEGER);
68 BEGIN
69   POSIT0(PX,PY);
70 PX:=(PX-XCO-XOS)*YSYM-XSH;
71 PY:=(PY-YCO-YOS)*XSYM-YSH
72 END;
73 PROCEDURE CURSOR(X,Y:INTEGER);
74 BEGIN
75   TRANS(X,Y);CURSOR0(X,Y)
76 END;
77 PROCEDURE PLOT(X,Y:INTEGER);
78 BEGIN
79   TRANS(X,Y);PLOT0(X,Y)
80 END;
81 PROCEDURE DRAW(DX,DY:INTEGER);
82 VAR X,Y:INTEGER;
83 BEGIN
84   POSIT(X,Y);X:=X+DX;Y:=Y+DY;
85   TRANS(X,Y);
86   DRAW0(DX*YSYM,DY*XSYM)
87 END;
88 PROCEDURE LINETO(X2,Y2:INTEGER);
89 VAR X1,Y1:INTEGER;
90 BEGIN
91   POSIT(X1,Y1);DRAW(X2-X1,Y2-Y1)
92 END;
93 PROCEDURE LINE(X1,Y1,X2,Y2:INTEGER);
94 BEGIN
95   PLOT(X1,Y1);DRAW(X2-X1,Y2-Y1)
96 END;
97 PROCEDURE CIRCLE(X,Y,R:INTEGER);

```



```

98 BEGIN
99 TRANS(X,Y);CIRCLE0(X,Y,R)
100 END;
101 PROCEDURE DTYPE(T:INTEGER);
102 BEGIN
103 GDTYPE:=T;
104 OVER(0);INVERSE(0);
105 IF T=0 THEN INVERSE(1);
106 IF T=2 THEN OVER(1)
107 END;
108 PROCEDURE PATTERN(PAT,RP:INTEGER);
109 BEGIN
110 GPAT:=PAT;GROT:=RP
111 END;
112 PROCEDURE CLS;
113 BEGIN
114 PAGE:AT(0,0);CURSOR(0,0);
115 INLINE(205,173,28,58,141,92,33,0,88);
116 INLINE(17,1,88,1,255,2,119,237,176)
117 END;
118 PROCEDURE GINIT;
119 BEGIN
120 XMIN:=0;YMIN:=0;XMAX:=255;YMAX:=175;
121 XOS:=0;YOS:=0;XCO:=0;YCO:=0;XSH:=0;YSH:=0;
122 XSYM:=1;YSYM:=1;
123 CURSOR(0,0);
124 DTYPE(1);
125 PATTERN(1,1)
126 END;
127 PROCEDURE WINDOW(X1,Y1,X2,Y2,X0,Y0:INTEGER);
128 BEGIN;
129 XMIN:=X1;YMIN:=Y1;XMAX:=X2;YMAX:=Y2;XOS:=X1+X0;
YOS:=Y1+Y0
130 END;
131 PROCEDURE COORD(X0,Y0:INTEGER);
132 BEGIN
133 XCO:=X0;YCO:=Y0
134 END;
135 PROCEDURE SYMMET(SX,SY:INTEGER);
136 BEGIN
137 IF SX=0 THEN XSYM:=1 ELSE XSYM:=-1;
138 IF SY=0 THEN YSYM:=1 ELSE YSYM:=-1
139 END;
140 PROCEDURE SHIFT(SX,SY:INTEGER);
141 BEGIN
142 XSH:=SX;YSH:=SY
143 END;
144 PROCEDURE CLW(FRAME:BOOLEAN);
145 VAR Y,T:INTEGER;
146 BEGIN
147 T:=GDTYPE;DTYPE(0);
148 FOR Y:=YMIN TO YMAX DO BEGIN;
149 PLOT0(XMIN,Y);DRAW0(XMAX-XMIN,0)
150 END;
151 DTYPE(T);
152 IF FRAME THEN BEGIN
153 CURSOR0(XMIN,YMIN);
154 DRAW0(0,YMAX-YMIN);DRAW0(XMAX-XMIN,0);
155 DRAW0(0,YMIN-YMAX);DRAW0(XMIN-XMAX,0)
156 END;
157 CURSOR(0,0)
158 END;
159 PROCEDURE TRIA(X1,Y1,X2,Y2,X3,Y3:INTEGER);
160 BEGIN
161 CURSOR(X1,Y1);
162 DRAW(X2-X1,Y2-Y1);
163 DRAW(X3-X2,Y3-Y2);
164 DRAW(X1-X3,Y1-Y3)
165 END;
166 PROCEDURE TETRA(X1,Y1,X2,Y2,X3,Y3,X4,Y4:INTEGER);
167 BEGIN
168 CURSOR(X1,Y1);
169 DRAW(X2-X1,Y2-Y1);
170 DRAW(X3-X2,Y3-Y2);
171 DRAW(X4-X3,Y4-Y3);
172 DRAW(X1-X4,Y1-Y4)
173 END;
174 PROCEDURE BOX(X1,Y1,X2,Y2:INTEGER);
175 BEGIN
176 CURSOR(X1,Y1);
177 DRAW(0,Y2-Y1);
178 DRAW(X2-X1,0);
179 DRAW(0,Y1-Y2);
180 DRAW(X1-X2,0)
181 END;
182 PROCEDURE RANGE(VAR X1,X2:INTEGER; Y:INTEGER);
183 BEGIN
184 IF X1>XMIN THEN
185 REPEAT X1:=X1-1
186 UNTIL (X1=XMIN) OR POINT0(X1,Y);
187 IF X2<XMAX THEN
188 REPEAT X2:=X2+1
189 UNTIL (X2=XMAX) OR POINT0(X2,Y)
190 END;
191 {S+}
192 PROCEDURE FILL0(VAR X:INTEGER;Y:INTEGER);
193 VAR X1,X2:INTEGER;
194 BEGIN
195 X2:=X;
196 CURSOR(X,X2,Y);
197 CURSOR0(X,Y);
198 X:=X+1;
199 DRAW0(X2-X,0);
200 X1:=X;
201 IF Y<YMAX THEN
202 WHILE X<X2 DO BEGIN
203 IF NOT POINT0(X,Y+1) THEN FILL0(X,Y+1);
204 X:=X+1
205 END;
206 X:=X1;
207 IF Y>YMIN THEN
208 WHILE X<X2 DO BEGIN
209 IF NOT POINT0(X,Y-1) THEN FILL0(X,Y-1);
210 X:=X+1
211 END;
212 X:=X2;
213 END;
214 {S-}
215 PROCEDURE FILL(X,Y:INTEGER);
216 VAR T,X1:INTEGER;
217 BEGIN
218 TRANS(X,Y);
219 IF NOT POINT0(X,Y) THEN BEGIN
220 T:=GDTYPE;DTYPE(1);
221 X1:=X;
222 FILL0(X1,Y);
223 DTYPE(T)

```

```

224 END;
225 CURSOR0(X,Y)
226 END;
227 {S+}
228 PROCEDURE PAINT0(X,Y,PAT:INTEGER);
229 VAR X1,X2,PAT1:INTEGER;
230 FUNCTION ROT(PAT,N:INTEGER):INTEGER;
231 VAR I:INTEGER;
232 BEGIN
233 FOR I:=-1 DOWNT0 N DO BEGIN
234 PAT:=PAT*2;
235 IF PAT>255 THEN PAT:=PAT-255
236 END;
237 FOR I:=1 TO N DO BEGIN
238 IF PAT MOD 2=1 THEN PAT:=PAT+256;
239 PAT:=PAT DIV 2
240 END;
241 ROT:=PAT
242 END;
243 BEGIN
244 X1:=X;X2:=X;
245 RANGE(X1,X2,Y);
246 X1:=X1+1;X2:=X2-1;
247 PAT:=ROT(PAT,(X-X1) MOD 8);
248 PAT1:=PAT;
249 FOR X:=X1 TO X2 DO BEGIN
250 PAT:=PAT*2;
251 IF PAT>255 THEN BEGIN
252 PAT:=PAT-255;
253 PLOT0(X,Y)
254 END
255 END;
256 IF Y<YMAX THEN BEGIN
257 PAT:=ROT(PAT1,XSYM*YSYM*GROT);
258 FOR X:=X1 TO X2 DO BEGIN
259 IF (PAT>127) AND NOT POINT0(X,Y+1) THEN
PAINT0(X,Y+1,PAT);
260 PAT:=PAT*2;
261 IF PAT>255 THEN PAT:=PAT-255
262 END;
263 END;
264 IF Y>YMIN THEN BEGIN
265 PAT:=ROT(PAT1,-XSYM*YSYM*GROT);
266 FOR X:=X1 TO X2 DO BEGIN
267 IF (PAT>127) AND NOT POINT0(X,Y-1) THEN
PAINT0(X,Y-1,PAT);
268 PAT:=PAT*2;
269 IF PAT>255 THEN PAT:=PAT-255
270 END;
271 END;
272 {S-}
273 {S+}
274 PROCEDURE PAINT(X,Y:INTEGER);
275 VAR T:INTEGER;
276 BEGIN
277 TRANS(X,Y);
278 IF NOT POINT0(X,Y) THEN BEGIN
279 T:=GDTYPE;DTYPE(1);
280 PAINT0(X,Y,GPAT);
281 DTYPE(T)
282 END;
283 CURSOR0(X,Y);
284 END;
285 PROCEDURE INK(I:INTEGER);
286 BEGIN
287 INLINE(55,221,86,2,205,52,34)
288 END;
289 PROCEDURE PAPER(P:INTEGER);
290 BEGIN
291 INLINE(167,221,86,2,205,52,34)
292 END;
293 PROCEDURE BRIGHT(B:INTEGER);
294 BEGIN
295 INLINE(151,221,86,2,205,116,34)
296 END;
297 PROCEDURE FLASH(F:INTEGER);
298 BEGIN
299 INLINE(175,60,221,86,2,205,29,34)
300 END;
301 PROCEDURE BORDER(B:INTEGER);
302 BEGIN
303 INLINE(221,126,2,205,151,34)
304 END;
305 {L+,0+,C+,S+,I-}

```

#### Výpis 4. Program Průběh funkce (926-V4)

```

PROGRAM PrubehFunkceSinus;
CONST
PI=3.14159;
VAR
S,X,U:REAL;

{#F grafR0.W}
{#F grafR1.W}

BEGIN
GINIT;
CLS;
WINDOW(0,0,255,191,16,96);
AT(2,20); { pro 64 znaků na řádek }
Writeln('PRUBEH FUNKCE Y=SIN(X)');
S:=90/PI;
SCALE(S,S);
LTYPE(3);
LINE(0,-1,0,1);
LINE(0,0,2*PI,0);
LTYPE(1);
PLOT(0,0);
U:=3/S;
X:=0;
WHILE X<=2*PI DO BEGIN
LINETO(X,SIN(X));
X:=X+U
END
END.

```



### Výpis 3. Program Schody (926-V3)

```

PROGRAM SCHODY;
VAR
{$F graf0.W}
{$F graf1.W}
{$F graf2.W}
{$F graf3.W}
{$F colour.W}

{$L-}
PROCEDURE SCHODY(N:INTEGER);
VAR I:INTEGER;
PROCEDURE SCHOD;
BEGIN
    TETRA(0,20,0,10,50,0,50,10);
    DRAW(20,4);
    DRAW(50,-10);
    DRAW(-20,-4);
    FILL(25,10);
    PAINT(35,17)
END;
BEGIN
    FOR I:=0 TO N-1 DO BEGIN
        SHIFT(20*I,14*I);
        SCHOD
    END;
    SHIFT(0,0);
    IF N>1 THEN BEGIN
        PLOT(50,0);
        DRAW(40,8);
        DRAW(20*(N-2),14*(N-2));
        DRAW(0,20)
    END
END;

{$L+}
BEGIN
    GINIT;
    PAPER(4);
    BORDER(4);
    CLS;
    SCALE(1.3,1.5);
    SCHODY(6);
    WINDOW(24,80,127,159,10,10);
    PAPER(5);
    CLW(TRUE);
    SCALE(0.4,0.4);
    PATTERN(17,-1);
    SCHODY(8);
    WINDOW(96,24,231,111,125,10);
    PAPER(7);
    CLW(TRUE);
    SYMMET(0,1);
    SCALE(0.6,0.6);
    PATTERN(85,1);
    SCHODY(7)
END.

```

### Výpis 5. Program Pneumatika (926-V5)

```

PROGRAM PNEUMATIKA;
CONST
    POSUN=15;
    R1=30;
    R2=65;
VAR
    I:INTEGER;
{$F graf10.W}
{$F graf11.W}
{$F graf13.W}

BEGIN
    GINIT;
    CLS;
    COORD(128,88);
    FOR I:=0 TO 1 DO BEGIN
        SHIFT(I*POSUN,I*POSUN);
        CIRCLE(0,0,R1);
        CIRCLE(0,0,R2);
        IF I=0 THEN
            FILL(R1+1,0)
    END
END.

```

## INTERFEJS NA SÉRIOVÉ PREPOJENIE MIKROPOČÍTAČA S TLAČIARŇOU

Jozef Kutej a Peter Kottáš, CHTF SVŠT, Radlinského 9, 812 37 Bratislava

V mikropočítačoch československej výroby sa stretávame takmer výlučne s paralelným prepojením mikropočítač-tlačiarne. Výhoda, ktorú toto prepojenie má – zvýšená rýchlosť prenosu dát, sa u mikropočítačov neprejaví. U nás bežne používané stolné tlačiarne D 100, PRT 80, K 6313, K 6314 majú rýchlosť tlače do 120 znakov/s. Týmto limitujú maximálnu rýchlosť prenosu dát na úroveň do 2000 bitov/s.

Paralelné prepojenia (systém PMD 85, Centronix, IRPR) vyžadujú na prepojenie minimálne jedenásť až dvanásť vodičov, obvod paralelného styku (aspoň 1/2 8255), budiče zbernice (8286, 8287 ...) a prípadne ďalšie hradlá na spracovanie riadiacich signálov. Zapojenie je obvodovo náročné, použité konektory, zvyčajne FRB, sú drahé a nedostatkové.

V našich mikropočítačoch sme odskúšali zapojenie sériového interfejsu na prepojenie mikropočítač-tlačiarne. S výhodou sa dá použiť v systémoch s mikroprocesorom MHB 8080A s tromi napájacími hladinami +12 V, +5 V a -5 V.

Interfejs na prepojenie mikropočítača s tlačiarňou vyžaduje tri tranzistory, tri diody a osem odporov. Na prepojenie sú potrebné tri vodiče. Tlačiarne K6313-14, D100, PRT80, môžu byť vybavené odpovedajúcim interfejsom RS 232 (V24), výrobcovia ich ponúkajú.

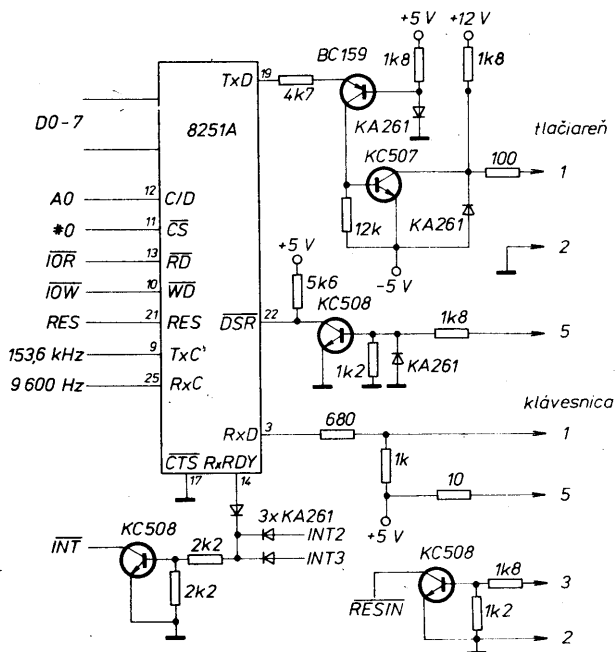
Na obr. 1 je príklad zapojenia obvodu styku mikropočítača PMD85 s tlačiarňou a klávesnicou. Tlačiareň a klávesnica majú sériové vstupy-výstupy, tlačiareň podľa normy RS232 (V24). Interfejs môže byť ovládaný cez SIO 8251 (obr. 1), SIO U856, cez dva vývody PIO 8255, U855 ...

Výhody zapojenia sú nasledovné:

- redukcia počtu prepojovacích vodičov z jedenásť na tri pri dĺžke prepojovacieho káblu do 16 m,
- zjednodušenie zapojenia obvodu styku, úspora polovodičov,
- na prepojenie nie je potrebné používať drahé a pri častejšom rozoberaní nespohľadlivé konektory FRB. Je možné použiť normalizované 25-kolíkové konektory. No nám sa osvedčili aj lacné 5-kolíkové nf-konektory s mechanickým spojením na závit,
- podstatné zníženie pracnosti pri výrobe prepojovacích káblov.

Tento interfejs je výhodné použiť v mikropočítačových systémoch s napájacími hladinami +12 V, +5 V a -5 V predovšetkým na prepojenie s tlačiarňou. Tu je ekonomicky aj technicky nevhodné realizovať interfejs pomocou IO 75150 a 75154, ktoré vyžadujú na správnu funkciu ďalšiu napájaciu hladinu -12 V.

Zapojenie sme odskúšali pri prepojení mikropočítača PMD85 s tlačiarňami K6311 (NDR) a D100 (PLR). Na 16 m sme prenášali dáta rýchlosťou 9600 bitov/s.



Obr. 1. Schéma zapojenia vstupov-výstupov pre tlačiareň a klávesnicu. Obvody INT a RESIN umožňujú prácu prerušovacieho systému a externý studený štart z externej klávesnice. (908-1)

# SOUŘADNICOVÝ ZAPISOVAČ ŘÍZENÝ MIKROPROCESOREM

Vladimír Julius, OK1IVJ, Sokolovská 123, 323 16 Plzeň

V minulém roce se v naší maloobchodní síti objevily souřadnicové zapisovače 1P16 k počítači SHARP MZ-800. Elektronika zapisovače obsahuje jednočipový mikroprocesor 8050 se 4kB paměti PROM a výstupní zesilovače pro ovládání krokových motorů a pera. Tyto zapisovače používají speciální papír v roli o šířce 11 cm, který není běžně na trhu, a potřeba per v ceně 80 Kčs (souprava) také nepatří k jejich přednostem. Proto jsem zkonstruoval zapisovač, který odstraňuje uvedené nedostatky a umožňuje připojení k libovolnému typu počítače. Používám ho ve spojení s počítačem SHARP MZ-821 a ZX SPECTRUM +.

V tomto příspěvku nebudu uvádět popis mechanické části zapisovače, její řešení je dnes dostatečně známé a neskrývá při pečlivém provedení žádné záludnosti. Pro posun papíru a pera se všeobecně používají krokové motorky, které se dají jednoduše řídit výstupními porty mikropočítače, jak bude popsáno v dalším textu. Krok zapisovače je převážně volen 0,125 nebo 0,25 mm, což umožňuje mimo jiné kreslení plošných spojů v rastru 2,5 mm. Doporučuji všem případným zájemcům, aby se seznámili s provedením zapisovačů ALFI a MINIGRAF O507.

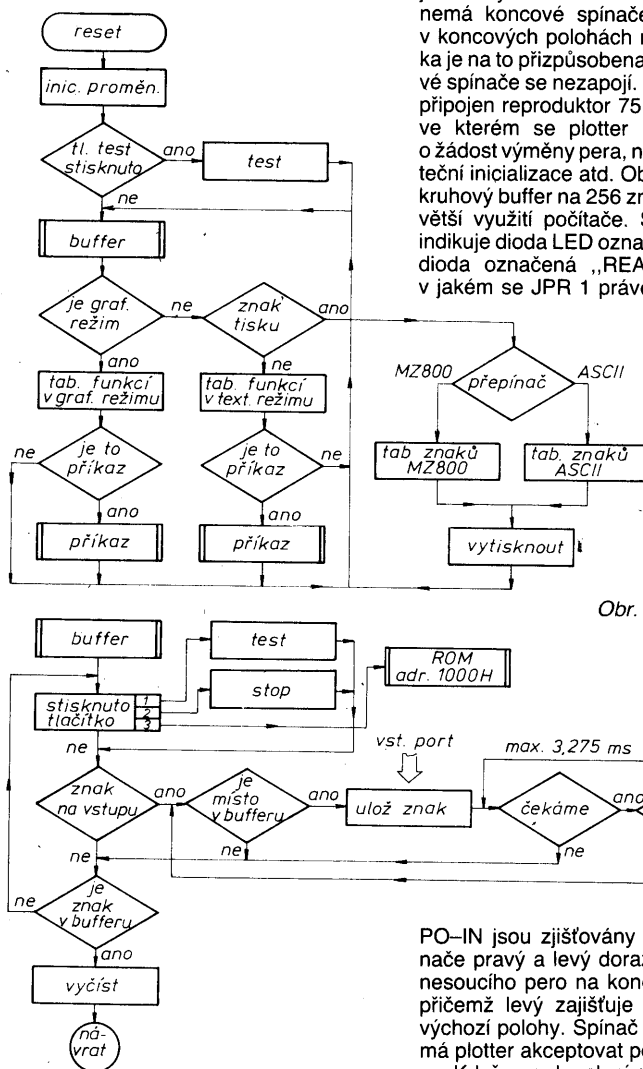
Řízení zapisovače je řešeno použitím procesorové desky JPR 1. Toto řešení není v dnešní době už jistě tím nejlepším, ale tuto procesorovou desku jsem měl k dispozici a ještě dnes se objevuje v inzerátech AR. Další verze je zpracována a odzkoušena s mikroprocesorem Z80, kde se tak podařilo snížit odběr o více než 50%. Zde jistě mnozí zájemci namítnou, proč jsme nepoužili jednočipový mikropočítač naší produkce např.

8035. Pokud ale budeme počítat, zjistíme, že nelze ušetřit žádné součástky a programování Z80 nám bylo bližší. Jiná situace by byla s použitím  $\mu P$  8250-4kB ROM... Na procesorové desce JPR 1 je osazeno 4 kB EPROM 2716 a 1 kB RAM 2114. Obsazení portů je patrné z Tab. 1. Port P1-IN je připojen na řídicí počítač a pomocí bitu RDA a RDP je řízen přenos mezi počítačem a procesorovou deskou JPR 1. Pro větší univerzálnost zařízení je vhodné do

řídících signálů RDA, STA, RDP, IRT zařadit hradla EX-OR MH7486, abychom mohli tyto signály invertovat. Signál STA udává status plotteru a je softwarově ošetřen a nevyužívá se. Signál IRT je použit pro reset JPR 1 od počítače. Port PO-OUT je použit pro řízení motorků a magnetu pro ovládání pera. Tento port je přímo slučitelný se vstupem plotteru MINIGRAF O507, se kterým byl také odzkoušen. Kdo má možnost použít tento plotter, nemusí si dělat starosti s mechanikou, která je zde vyřešena velice dobře. Tento plotter nemá koncové spínače, takže se motory v koncových polohách nezastaví; mechanika je na to přizpůsobena. Vývody pro koncové spínače se nezapojí. Přes vývod K3-28 je připojen reproduktor 75  $\Omega$ ; signalizuje stav, ve kterém se plotter nachází. Jde např. o žádost výměny pera, najetí na doraz, počáteční inicializace atd. Obslužný program má kruhový buffer na 256 znaků a tak umožňuje větší využití počítače. Stav tohoto bufferu indikuje dioda LED označená „BUFF“. Další dioda označená „READY“ indikuje stav, v jakém se JPR 1 právě nachází. Na portu

Tab. 1 Obsazení portů JPR - 1

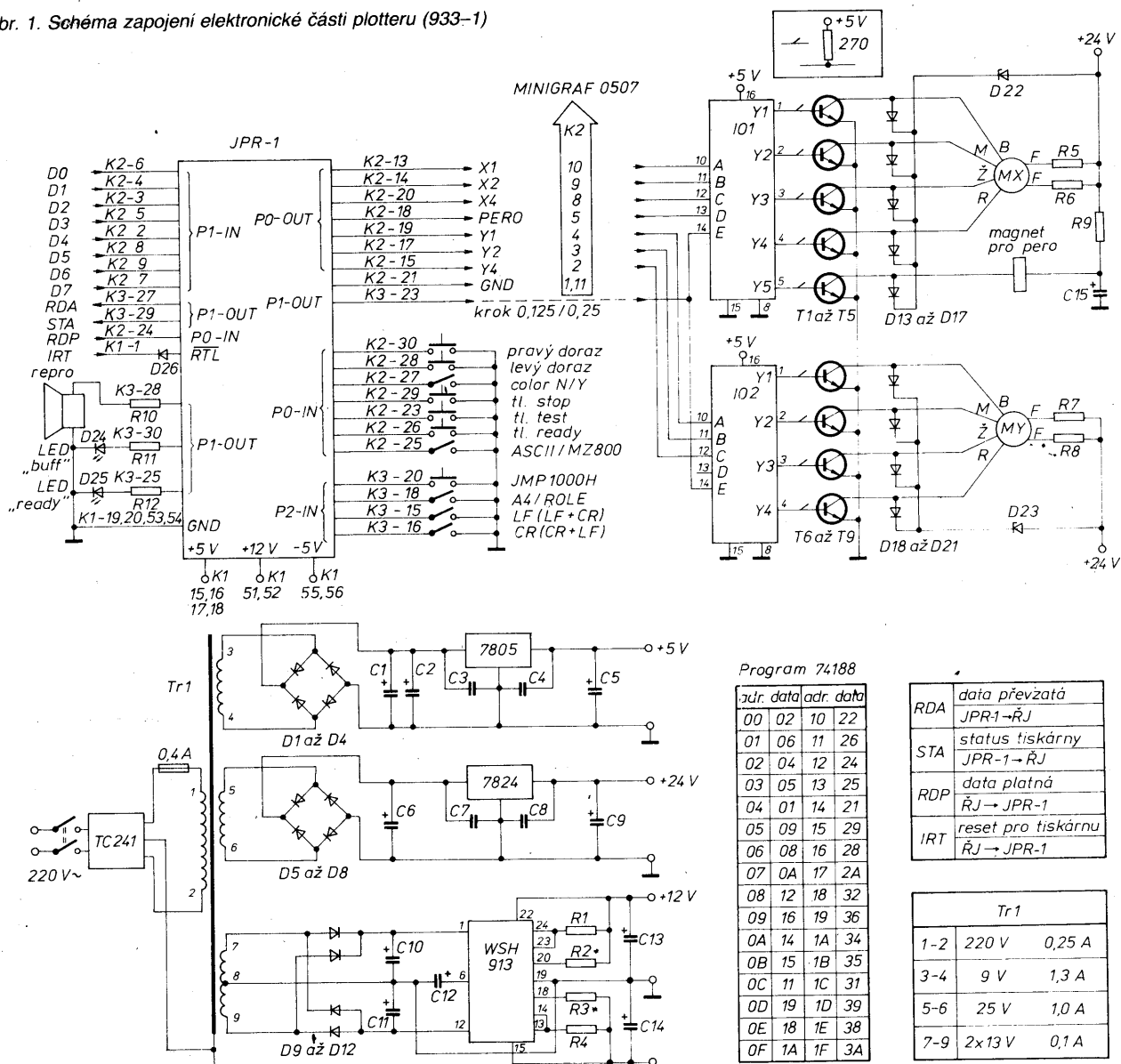
PORT - vst. bit	KONEKTOR	SIGNÁL	H > L
RDSI1	0	K3 - 20	JMP 1000H
	1	K3 - 17	-
	2	K3 - 12	-
	3	K3 - 14	-
	4	K3 - 13	-
	5	K3 - 18	A4 / ROLE
	6	K3 - 15	LF (LF + CR)
RDSI	7	K3 - 16	CR (CR + LF)
	0	K2 - 30	PRAVÝ DORAZ
	1	K2 - 28	LEVÝ DORAZ
	2	K2 - 27	COLOR A/N
	3	K2 - 29	TL. STOP
	4	K2 - 23	TL. TEST
	5	K2 - 26	TL. READY
DSHI	6	K2 - 24	RDP (DATA PLATNÁ)
	7	K2 - 25	ASCII / MZ 800
	0	K2 - 6	DATA 0
	1	K2 - 4	1
	2	K2 - 3	2
	3	K2 - 5	3
	4	K2 - 2	4
PORT - výst. bit	5	K2 - 8	5
	6	K2 - 9	6
	7	K2 - 7	7
PORT - výst. bit	KONEKTOR	SIGNÁL	
RDSO	0	K3 - 24	-
	1	K3 - 26	-
	2	K3 - 27	RDA (DATA AKCEPT)
	3	K3 - 29	STA (STATUS TISK.)
	4	K3 - 23	KROK 0.125/0.25
	5	K3 - 28	REPRODUKTOR
	6	K3 - 30	LED "BUFF"
MOTR	7	K3 - 25	LED "READY"
	0	K2 - 13	PÁZE - X 1
	1	K2 - 14	X 2
	2	K2 - 20	X 4
	3	K2 - 19	Y 1
	4	K2 - 17	Y 2
	5	K2 - 15	Y 4
	6	K2 - 16	-
	7	K2 - 18	OVL. PERA



Obr. 2 Grafické schéma programu pro plotter (933-2)

PO-IN jsou zjišťovány stavy spínačů. Spínače pravý a levý doraz hlásí najetí vozíku nesoucího pero na konec kreslicího stolku, přičemž levý zajišťuje inicializaci pera do výchozí polohy. Spínač „COLOR“ říká, zda má plotter akceptovat povel pro změnu barvy. Když ano, kreslení se zastaví a čeká se

Obr. 1. Schéma zapojení elektronické části plotteru (933-1)



na výměnu pera. Z reproduktoru se ozývá přerušovaný tón signalizující barvu pera, kterou si počítač žádá. Jde o sled 1-4 teček. Stiskem tlačítka „READY“ se plotter opět spustí. Při rozeptnutí spínače „COLOR“ máme možnost kreslit barevný obrazec pouze jednou barvou bez přerušení. Tlačítko „STOP“ umožňuje okamžité zastavení plotteru – spustí se opět tlačítkem „READY“. Protože tento plotter byl řešen hlavně pro použití s počítačem SHARP, který má pro malá písmena abecedy jiné kódy než udává tabulka ASCII, je nutné tyto znaky převádět. To je zajištěno s přepínačem přepnutým do polohy MZ-800. Pokud je tento přepínač v poloze ASCII, znaky se tisknou podle této tabulky. Znaky, které nejsou v generátoru znaků obsaženy, jsou nahrazeny tečkou. Tlačítko „JMP 1000H“ vykoná tuto instrukci a provede nepodmíněný skok na adresu 1000 pro případné další využití JPR1. Přepínač „A4/ROLE“ sdělí počítači, jestli máme založen list papíru A4 a pro další psaní musíme inicializovat znovu začátek stránky, nebo zda máme roli a stačí vynechat několik volných řádků. K přepínačům „LF“ a „CR“ jistě není co dodávat. Ještě je třeba se zmínit o výstupu pro velikost kroku 0,125 nebo 0,25 mm. Tento výstup vybírá dekodér naprogramovaný v paměti MH74188. Pokud bude mít někdo připojen plotter 0507, potom nemůže tuto funkci využít a vývod zůstane nezapojen.

Ovládání krokových motorů je řízeno tříbitovým kódem 0-7, který přes dekodér ovládá 4 fáze krokového motoru. Tento dekodér v režimu 0,125 mm otáčí motorem o 4,5 stupně, což představuje 80 kroků na jednu otáčku. V režimu 0,25 mm se motor otáčí o 9 stupňů na jeden krok. Úměrně tomu je také softwarově zmenšena rychlost krokování tak, aby výsledná rychlost psaní zůstala nezměněna. To platí při použití dekodéru MH74188, který je pro tento účel naprogramován podle tabulky. Řešení s dekodérem bylo použito vzhledem k jednoduchému připojení na plotter MINIGRAF 0507, který obdobný dekodér obsahuje. V další verzi programu je tento dekodér řešen softwarově, což dále zjednodušuje elektroniku plotteru.

V plotteru jsou použity motory SMR 300/300 RI24, které se dají objednat pro organizaci v MEZ Náchod. V maloobchodní síti se dají občas koupit motory SMR 300/100, které pracují stejně, pouze výsledná rychlost se bude muset v programu zmenšit asi o 1/3.

Dále popsany program řadí tento plotter mezi inteligentní periférie, které na našem trhu stále nejsou a zřejmě ještě dlouho nebudou v takových cenových relacích, aby byly dostupné uživatelům domácích počítačů. Máme tak vyřešen grafický výstup počítače a jsme schopni kreslit technické výkresy, výkresy plošných spojů a třeba diplomy nebo prání k svátku.

## Seznam součástek

Rezistory:		
R1, R4	6,8 Ω	TR 191
R2*	12 kΩ	TR 191
R3*	8,2 kΩ	TR 191
R5-R8	68 Ω	TR 510
R9	47 Ω	TR 510
R10-R12	100 Ω	TR 191

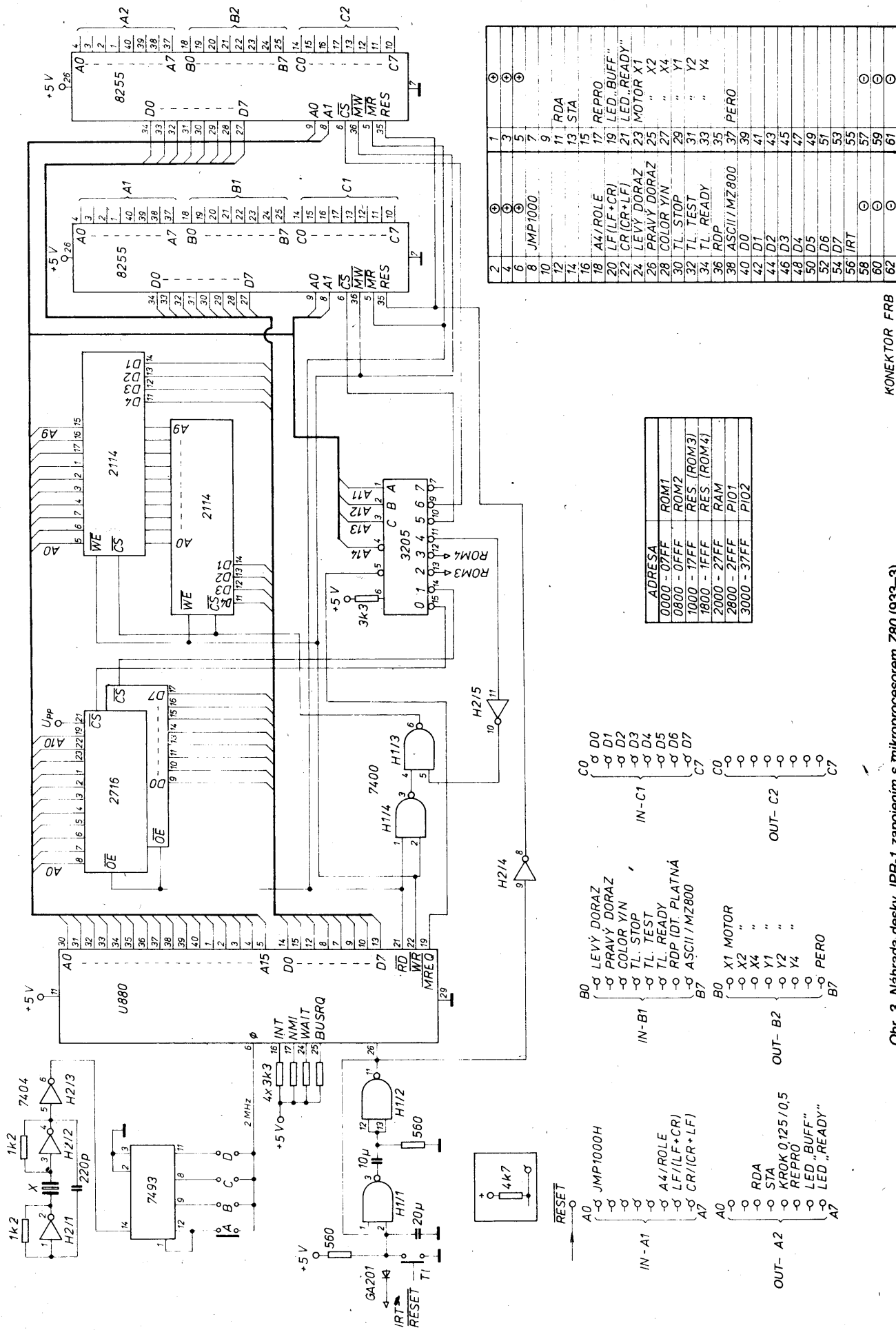
Kondenzátory:		
C1, C2, C5	1 GF	TE 984
C3, C4, C7, C8	100 nF	TK 782
C6, C15	500 nF	TE 986
C9	50 nF	TE 986
C10, C11	200 nF	TE 986
C12-C14	4,7 nF	TE 125

Polovodiče:		
D1-D8	1N5401	
D9-D12	KY130/80	
D13-D21	KY130/150	
D22, D23	KZ260/18	
D24	LQ1702	
D25	LQ1202	
D26	GA204	
T1-T9	KF508	
I01, I02	MH74188	
I03	MH7805	
I04	MH7824	
I05	WSH913	

Ostatní:  
MX, MY  
REPRO ARZ 084

SMR 300/300 RI24

\* Nutno nastavit výstupní napětí.



Obr. 3. Náhrada desky JPR-1 zapojením s mikroprocesorem Z80 (933-3)

```

;*****
;*
;* Program pro rizeni plotteru z ridici jednotky
;*   pres standartni rozhrani Centronics
;*
;* Julius Roman Sokolovska 123 Plzen 323 16
;*
;*****

```

```

.8000      ;soubor instrukci 18000
.RADIX 16  ;implicitni ciselna soustava

```

```

SYKORA EQU 1000 ;zacatek dalsich 4 kbyte
NEWP EQU 8000 ;zde zacina vlastni program
;JPR-1 EQU 0
RAM EQU 0B000 ;pocatecni adresa promennych
;JPR-1 EQU 2000
RDSI1 EQU 8E ;vstup ridicich signalu
;JPR-1 EQU 2C00
RDSI EQU 90 ;vstup ridicich signalu
;JPR-1 EQU 2400
RDSO EQU 8C ;vystup ridicich signalu
;JPR-1 EQU 2800
DSHI EQU 8D ;vstup dat
;JPR-1 EQU 2800
MOTR EQU 90 ;vystup na krokove motory
;JPR-1 EQU 2400

```

```

.PHASE NEWP
DI ;zakaz preruseni
LXI SP,STACK ;a nastav stack
MVI A,4 ;pocatecni nastaveni
STA RDSO ;vystupniho portu
XRA A ;znuluj
STA KLM ;bit reproduktoru
STA PORG ;funkci dorazu pro ORG
STA KR025 ;krok- 0.125 mm
LDA RDSI ;otestuj tlacitko
ANI 10 ;TEST, je-li stisknuto
CZ TEST ;proved celkovy test
CALL MUS1 ;piskej
CALL GENINI ;inicializace generatoru

```

```

RS1: XRA A ;znuluj
STA KLM ;bit reproduktoru
STA UKRA ;ukazatel radek
STA UKZR ; - " - znaku na radce
STA KR025 ;jednoduchy krok 0.125 mm
STA RDSO ;ridici signaly
STA SPD ;rychlost
STA PORG ;funkci dorazu pro ORG
INR A ;barva=1
STA COLR
MVI A,3F ;pocet radek na stranku
STA PRST
CALL BUFINI ;inicializace bufferu
LXI B,0
LXI D,0 ;parametry pro ORG
CALL PORG ;nastav pocatek sour. syst.
CALL CH400 ;80 znaku na radek
LXI H,0 ;skutecna X=0
SHLD SKUTX
MVI A,1
STA REZI ;textovy rezim
STA ASCD ;rezim ASCII
STA PORG ;funkce dorazu pro havarii

```

```

RIBL: CALL BUFE ;cyklus pro vycitani prikazu
LXI D,RIBL ;navratova adresa pro pprog.
PUSH D ;do stacku
LDA REZI
CPI 2 ;podle druhu rezimu vyber tab.
LXI H,TBFG ;graficky-graficka tabulka
JZ RIBG ;a vyber
MOV A,C ;rezim textovy
CPI 1D ;je-li kod 1DH
JZ RIB1 ;vyber adresu
ANI 0F0 ;je-li vetsi nez 0FH ->
JNZ PRINT ;tisknuteiny znak - vytiskni

```

```

RIB1: LDA RDSI ;ridici znak
ANI 80
CZ PREVOD ;prevod podle prepinnace
LXI H,TBFT ;textova tabulka

```

```

RIBG: CALL PROHL ;vyber podprogram z tabulky
PCHL ;a skoc na nej

```

```

PROHL: MOV A,M ;u C je povel, u HL adresa tab.
CMP C ;vybere adresu
JZ PROHU ;nasel prikaz
CPI 0FF ;konec tabulky
JZ PROHK
INX H ;HL=HL+3 ->
INX H ;dalsi pozice
INX H
JMP PROHL ;dalsi prikaz v tabulce

```

```

PROHU: INX H ;vybere z tabulky adresu
MOV E,M ;do E
INX H
MOV D,M ;dalsi pulku do D
XCHG ;a vumen s HL
RET ;navrat

```

```

PROHK: POP H ;povel jsem nenasel
POP H ;zrus navratovou adresu
JMP RIBL ;dalsi povel

```

```

HAVAR: LXI SP,STACK ;zjistil jsem doraz
CALL MUS2 ;a hlasim havarii
JMP RS1 ;restartuj system

```

```

TBFG: DB 48 ;tabulka funkci
DW HOME ;u grafickem rezimu
DB 54
DW SETS
DB 45
DW CGRAMS
DB 47
DW CGRAMW
DB 43
DW COLOR
DB 4A
DW RLINE
DB 52
DW RMOVE
DB 4C
DW LTYPE
DB 44
DW DRAW
DB 49
DW HSET
DB 4D
DW PMOVE
DB 58
DW AXIS
DB 53
DW SCALE
DB 51
DW ANGLE
DB 50
DW GPRINT
DB 1
DW TEXTR
DB 46
DW FAST
DB 5A
DW SSIZE
DB 59
DW DOT
DB 57
DW RDOT
DB 42
DW DIAC
DB 0FF
DW PROHK

```

```

TBFT: DB 2 ;funkce v textovem rezimu
DW GRAFR
DB 5
DW CGRAMW
DB 0D
DW PCR
DB 0A
DW PLF
DB 8
DW BASP
DB 3
DW LINUP
DB 4
DW PTEST
DB 9
DW DEVET
DB 0B
DW CH26
DB 7
DW CH400
DB 1
DW TEXTT
DB 0C
DW STRAN
DB 1D
DW CHCOL
DB 0FF
DW PROHK

```

```

BUFINI: XRA A ;inicializace bufferu
LXI H,CW ;znuluj 1. ukazovatel
MOV M,A
INX H
MOV M,A ;pote i druhe
STA CK
STA BUFINT
RET

```

```

BUFE: PUSH PSW ;pprog. pro rizeni prijmu

```

	PUSH	D	; dat a prace s bufferem		MOV	E, M	
	PUSH	H			XCHG		
	LDA	RDSI	; test tl. STOP		INX	H	
	ANI	8			DAD	D	
	CZ	MUS3	; stisknuto-pozdrz praci		MOV	M, C	
	LDA	RDSI	; test tl. TEST		XCHG		
	ANI	10			INR	M	
	CZ	PTEST	; stisknuto-proved test		RNZ		
	LDA	RDSI1	; test tl. SYKORA		LXI	H, CK	
	ANI	1			MOV	A, M	
	JZ	SYKORA	; stisknuto-skoc na jeho adr.		RAR		
	CALL	BUFO	; ulozeni znaku z portu		CMC		
	LDA	BUFINT	; znak cten z buff.		RAL		
	ORA	A	; nebo z pomocne promenne	BFO3:	MOV	M, A	
	JNZ	NEZB	; -z promenne		INR	A	
	CALL	BFI	; -z bufferu		RET		
	PUSH	PSW	; uschovej				
	LDA	KR025		PARAM:	CALL	IGNO	; precteni ciselnych param.
	ORI	40			MOV	A, C	; z bufferu
	STA	RDSO	; rozsvit diodu READY		CPI	20	; test na zaporna cisla
	POP	PSW			MVI	A, 0	; kladne
	JZ	BUFE+3	; buff. prazdny - LED sviti		JNZ	KL1	
	LDA	KR025			MVI	A, 1	; zaporne
BUFZ:	STA	RDSO	; neni, zhasni ji	KLAD:	STA	PRZ	; nastav priznak pro pozdejsi
	POP	H			LXI	H, 0	; invertovani
	POP	D			CALL	EX0	; vlastni cteni
	POP	PSW			JNC	COMP	; zinvertovani podle znamenka
	RET		; obnov reg. a vrat se		MOV	A, C	; uloz posledni znak znovu
					STA	PRECH	; do bufferu
NEZB:	XRA	A	; znak neni z bufferu		MVI	A, 1	
	STA	BUFINT	; priste uz bude		STA	BUFINT	
	LDA	PRECH	; nacti jej do C				
	MOV	C, A		COMP:	LDA	PRZ	; je-li tato prom. nastavena
	JMP	BUFZ	; a vrat se		CPI	1	; na 1 neprovadi inventaci HL
					RNZ		
BUFO:	LDA	RDSI	; jsou pripravena data ?		MOV	A, L	; nejprve L
	ANI	40			CMA		
	RZ		; ne, vrat se		MOV	L, A	
	LDA	DSHI	; ano, cti je do C		MOV	A, H	; pote i H
	MOV	C, A			CMA		
	CALL	BFO	; pokus se je ulozit do buff.		MOV	H, A	
	RZ		; nejde to - je plny		INX	H	; HL=HL+1
	LDA	KR025			RET		; zinvertovane HL
	ORI	4		KL1:	MVI	A, 1	; neni-li 1. nemezerovy znak
	STA	RDSO	; potvrd prijem data		STA	BUFINT	; "-" uloz jej zpet do buff.
SEST:	LDA	RDSI	; cekej na shozeni RDP		MOV	A, C	
	ANI	40			STA	PRECH	
	JNZ	SEST			JMP	KLAD	; cislo je kladne
	LDA	KR025					
	STA	RDSO	; nastav znovu pripravenost	EX0:	CALL	BUFE	; vlastni cteni vstup HL=0
	MVI	B, 83			MOV	A, C	; vystup cislo v HL
BUFW:	DCR	B	; chvili pockej		CPI	3A	
	RZ		; nejdou, vratime se		CMC		
	LDA	RDSI	; a testuj prijdou-li		RC		; znak mimo rozsah vrat se
	ANI	40	; dalsi data	EX1:	MOV	B, A	
	JNZ	BUFO	; prisli		CALL	NIBBLE	
	JMP	BUFW	; cekej dale		RC		
					DAD	H	; zvys HL o dalsi cislo
BFI:	LXI	H, CK	; vlastni cteni z bufferu		PUSH	H	
	MOV	A, M			DAD	H	
	RAR				DAD	H	
	XRA	M			DAD	H	
	ANI	1			POP	D	
	LXI	H, BUFF-2			DAD	D	
	MOV	A, M			MOV	E, A	
	MOV	E, A			MVI	D, 0	
	INX	H			DAD	D	
	JNZ	BFI2			JMP	EX0	
	CMP	M		NIBBLE:	SUI	30	; testuj prislusnost
	RZ				RC		; do mnoziny
BFI2:	MVI	D, 0			ADI	30-47	; hexadecimalnich znaku
	XCHG				RC		; nepatri tam
	INX	H			ADI	6	
	DAD	D			JP	N10	
	MOV	C, M			ADI	7	
	XCHG				RC		; nepatri tam
	DCX	H		N10:	ADI	0A	
	INR	M			ORA	A	
	RNZ				RET		
	LXI	H, CK		IGNO:	CALL	BUFE	; ignoruj mezery
	MOV	A, M			MVI	A, 20	
	RAR				CMP	C	
	RAR				JZ	IGNO	
	CMC				RET		
	RAL			HSET:	LHLD	SKUTX	; funkce HSET
	RAL				XCHG		
	JMP	BFO3			LHLD	STPX	; nastav novy skut. pocatek
					DAD	D	
BFO:	LXI	H, CK	; vlastni zapis do bufferu		SHLD	SKUTX	
	MOV	A, M			JMP	NULXY	; a nuluj okamzite X a Y
	RAR			HOME:	LXI	B, 0	; funkce HOME
	XRA	M			LXI	D, 0	
	ANI	1			JMP	MOVA	; presun na 0, 0
	LXI	H, BUFF-2		COLOR:	CALL	PARAM	; funkce COLOR
	MOV	A, M			INR	L	
	INX	H					
	JZ	BFO2					
	CMP	M					
	RZ						
BFO2:	MVI	D, 0					

```

COL1:  MOV  A,L
      STA  CKLO
      STA  COLR
      LDA  RDSI
      ANI  4
      RNZ  ;barva je zakazana, vrat se
      LXI  H,000 ;neni, inicializuj zvukovy
      SHLD KLOPA ;generator
      LXI  H,800 ;podle cisla barvy
      SHLD KLOPB
      MVI  A,15
      STA  KMIT
      JMP  MUSIC ;hlas barvu

```

;Umoznuje nadefinovat vlastni znak

```

CGRAMS: CALL  PARAM
      MOV  A,L ;cislo generatoru
      ORA  A
      JNZ  CGRA
      INR  A
      ANI  0F ;uvazuje pouze 15 generatoru
      LXI  H,CGR-22;adresa generatoru
      LXI  D,22 ;maximalni delka
      CALL NASOB ;vypocit vyslednou adr.
      PUSH H ;uschovej ji
      MVI  E,1F ;pocet znaku v generatoru
      INX  H ;vyhrad byte pro delku
      XRA  A ;uvodni byte nastav
      MOV  M,A ;na nulu
      INX  H ;dalsi pozice
      CALL BUFE ;oddelovac pryc
      CGRS1: CALL BUFE ;znak z bufferu
      MOV  A,C
      MOV  M,A ;do generatoru
      INX  HL ;dalsi pozice
      DCR  E ;sniz zbyvajici pocet
      JZ   CGRSK ;znaku, je-li 0 ukonci to
      ANI  80 ;je-li 7. bit nastaven
      JZ   CGRS1 ;konec tvoreni znaku
      MVI  A,20 ;vypociti delku
      SUB  E
      POP  H ;a uloz ji
      MOV  M,A
      RET
      CGRSK: ORI  80 ;vycerpána povolena delka
      DCX  HL ;nastav 7. bit na jednicku
      MOV  M,A ;uloz to
      MVI  A,21 ;nastav maximalni
      POP  H ;delku
      MOV  M,A
      RET

```

;Vytisknuti preddefinovaneho znaku

```

CGRAMW: CALL  PARAM ;cislo generatoru
      MOV  A,L
      ORA  A
      JNZ  CGRC
      INR  A
      ANI  0F ;uvazuje pouze 15 generatoru
      LXI  H,CGR-22;adresa generatoru
      LXI  D,22
      CALL NASOB ;vypociti adresu
      MOV  A,M ;testuj delku
      ORA  A
      RZ   ;vrat se je-li 0
      MOV  E,A
      INX  H
      JMP  WRITE ;jdi tisknout
      GENINI: LXI  H,CGR ;inicializuje generator
      LXI  D,1FF ;v delce 1/2 kbyte
      GENIN1: XRA  A
      MOV  M,A
      INX  H
      DCX  D
      MOV  A,E
      ORA  D
      JNZ  GENIN1 ;pokracuj
      RET
      PMOVE: CALL  SKY ;funkce PMOVE
      JMP  MOVA
      DOT:  CALL  SKY ;funkce POINT
      JMP  POINTA
      RDOT: CALL  SKY ;funkce RPOINT
      JMP  POINTR
      SSIZE: CALL  PARAM ;funkce SIZE
      SHLD SIXX ;XX
      CALL  BUFE
      CALL  PARAM
      SHLD SIXY ;XY
      CALL  BUFE
      CALL  PARAM
      SHLD SIYX ;YX
      CALL  BUFE
      CALL  PARAM

```

```

SHLD SIYV ;YV
RET

```

```

DRAW:  CALL  RYCH ;funkce DRAW
      CALL  SKY
      CALL  VECTA
      JMP  RYCHN

```

```

LTYPE: CALL  PARAM ;funkce LTYPE
      MOV  A,L
      RLC
      RLC
      STA  TPLI ;parametr nasob 4
      STA  AKLI ;uloz jej
      STA  RTLI
      MVI  A,80
      STA  PSLI
      STA  STWP
      RET

```

```

SETS:  CALL  PARAM ;funkce SPEED
      MOV  A,L
      STA  SPD
      JMP  RYCHN

```

```

RYCH:  LDA  RTLI ;prepocte rychlost
      CPI  0 ;na skutecnou
      JZ   RYCHN ;podle delky kroku
      STA  TPLI ;a typu cary
      LDA  STWP
      STA  PSLI
      MVI  B,18
      CALL SPEDA
      RET

```

```

RYCHN: MVI  B,0B ;nastav rychlost
      CALL SPEDA ;pro presuny
      XRA  A
      STA  TPLI
      LDA  PSLI
      STA  STWP
      MVI  A,80
      STA  PSLI
      RET

```

```

SPEDA: LDA  SPD
      ADD  B
      STA  SPED
      RET

```

```

RMOVE: CALL  SKY ;funkce RMOVE
      JMP  MOVR

```

```

RLINE: CALL  RYCH ;funkce RLINE
      CALL  SKY
      CALL  VECTR
      JMP  RYCHN

```

```

SKY:  CALL  PARAM ;nacte z bufferu
      PUSH H ;souradnice X a Y
      CALL BUFE
      CALL PARAM
      MOV  B,H ;do BC
      MOV  C,L
      POP  D ;a DE
      RET

```

```

AXIS: CALL  BUFE ;funkce AXIS
      MOV  A,C
      CPI  31 ;osa do smeru
      JZ   AXISX ;X
      CALL AXIA ;parametry osy
      LXI  B,0
      LXI  D,0FFFFC
      CALL VECTR ;dilek
      LXI  B,0
      LXI  D,0
      CALL VECTR
      LXI  D,0FFFFC
      LXI  B,0
      CALL VECTR
      LXI  D,0
      LHL D
      MOV  B,H
      MOV  C,L
      CALL VECTR ;na dalsi pozici
      LDA  REPE ;sniz pocet opakovani
      DCR  A
      JZ   AXVK ;konec osy
      STA  REPE ;jeste ne
      JMP  DILY

```

```

DILY: LXI  D,0FFFFC
      CALL VECTR ;dilek
      LXI  B,0
      LXI  D,0
      CALL VECTR
      LXI  D,0FFFFC
      LXI  B,0
      CALL VECTR
      LXI  D,0
      LHL D
      MOV  B,H
      MOV  C,L
      CALL VECTR ;na dalsi pozici
      LDA  REPE ;sniz pocet opakovani
      DCR  A
      JZ   AXVK ;konec osy
      STA  REPE ;jeste ne
      JMP  DILY
      AXISX: CALL  AXIA ;parametry osy
      DILX:  LXI  B,0FFFFC ;dilek
      LXI  D,0
      CALL VECTR
      LXI  B,0
      LXI  D,0
      CALL VECTR
      LXI  D,0FFFFC
      LXI  B,0
      CALL VECTR
      LXI  D,0
      LHL D
      MOV  B,H
      MOV  C,L
      CALL VECTR ;na dalsi pozici
      LDA  REPE ;sniz pocet opakovani
      DCR  A
      JZ   AXVK ;konec osy
      STA  REPE ;jeste ne
      JMP  DILY

```

```

LHLD PITCH
XCHG
CALL VECTR ;na dalsi pozici
LDA REPE ;sniz pocet opakovani
DCR A
JZ AXKK ;koniec osy
STA REPE ;jeste ne
JMP DILX

AXIA: CALL BUFE ;nacte parametry osy
CALL PARAM
SHLD PITCH ;roztec
CALL BUFE
CALL PARAM
MOV A,L
STA REPE ;pocet dilku
RET

AXKK: LXI D,0 ;posledni dilek
LXI B,0FFFC
CALL VECTR
LXI D,0
LXI B,0
CALL VECTR
LXI D,0
LXI B,0FFFC
JMP VECTR
LXI B,0
LXI D,0FFFC
CALL VECTR
LXI B,0
LXI D,0
CALL VECTR
LXI B,0
LXI D,0FFFC
JMP VECTR

GPRINT: CALL BUFE ;funkce GPRINT
MOV A,C
CPI 0D ;je-li CR znamena to konec
RZ ;retezce
LDA RDSI
ANI 80
CZ PREVOD ;prevede znak podle prep.
MOV A,C
ANI 80
MOV A,C ;je-li stalem vetsi nez 127
JNZ GTEC ;vytiskni tecku
CALL PCHR ;vytiskni znak
JMP GPRINT ;znovu
MVI A,2E ;tecka
CALL PCHR ;vytiskni
JMP GPRINT ;znovu

GTEC:

NASOB: DAD D ;A krat pricte DE k HL
DCR A
JNZ NASOB
RET

ANGLE: CALL PARAM ;funkce ANGLE
MOV A,L ;uhel do A
CPI 0
JZ VODP ;vodorovne vpravo
CPI 1
JZ DOL ;svisle dolu
CPI 2
JZ VODL ;vodorovne vlevo
CPI 3
JZ NAH ;svisle nahoru
RET ;neznam

VODP: CALL VISI
SHLD SIXX ;XX,YY = velikost
SHLD SIYY
LXI H,0
SHLD SIXY ;XY,YX = 0
SHLD SIYX
RET

DOL: CALL VISI
SHLD SIYX ;YX = velikost
MVI A,1
STA PRZ
CALL COMP
SHLD SIXY ;XY = -velikost
LXI H,0
SHLD SIXX ;XX,YY = 0
SHLD SIYY
RET

NAH: CALL VISI
SHLD SIXY ;XY = velikost
MVI A,1
STA PRZ
CALL COMP
SHLD SIYX ;YX = -velikost
LXI H,0
SHLD SIXX ;XX,YY = 0
SHLD SIYY
RET

VODL: CALL VISI
MVI A,1
STA PRZ

```

```

CALL COMP
SHLD SIXX
SHLD SIYY ;XX,YY = -velikost
LXI H,0
SHLD SIYX
SHLD SIXY ;XY,YX = 0
RET

VISI: LDA SIZE ;podle SIZE vypocte
INR A ;do HL skutecnou velikost
MVI H,0
MOV L,A
RET

DIAC: CALL PARAM ;nastavi nebo znusi
MOV A,L ;diakriticky rezim
STA ASCD
RET

SCALE: CALL PARAM ;nastavi velikost znaku
MOV A,L
STA SIZE
MVI A,0 ;a uhel tisku 0
JMP ANGLE+4

PTEST: LDA REZI ;neni-li nastaven textovy
CPI 1 ;rezim nastav ho
CNZ TEXTR
CALL CR
CALL LF
CALL CH26
LXI H,TSTTB ;vytiskni text
MVI A,0E ;podle tabulky
TSTT: PUSH PSW
PUSH H
MOV C,M
CALL PRINT
POP H
INX
POP PSW
DCR A
JNZ TSTT
CALL CR
CALL LF
CALL MONOP
MVI A,4
PUSH PSW ;a podtrhni ho
LXI B,0
LXI D,1EF
CALL VECTR
LXI B,0FFFE
LXI D,0FE11
CALL MOVR
POP PSW
DCR A
JNZ PTES2
CALL LF
JMP TEXTT
MVI L,1 ;velikost znaku 1
CALL SCALE+3
MVI L,0 ;typ cary 0
CALL LTYP+3
MVI A,2 ;graficky rezim
STA REZI
CALL MONOP ;presun na zacatek
LHLD STPX ;inicializuj
SHLD SKUTX ;souradnice
CALL NULXY
RET

TEXTT: CALL CH408
JMP CR

TEXTR: MVI L,0 ;typ cary 0
CALL LTYP+3
MVI A,1 ;textovy rezim
STA REZI
STA ASCD ;ASCII rezim
LHLD SKUTX
XCHG
LHLD STPX
DAD D
SHLD STPX ;vypocti skutecnou X
LXI H,0
SHLD NOPX
SHLD SIXY ;parametry pro znak
SHLD SIXX ;80 znaku na radek
CALL CH408
XRA A
STA UKRA
STA UK2R ;znuluj ukazatele
CALL FAST+4 ;jednoduchy krok
RET

DEVET: CALL BUFE ;prisel kod $09
MOV A,C ;cti dalsi znak
CPI 9
JZ DEV1 ;je to opet $09
STA PRECH ;neni - vrat jej
MVI A,1 ;do bufferu
STA BUFINT
RET ;a zpet

DEV1: CALL BUFE ;dalsi znak
MOV A,C
CPI 9
JZ CH80 ;zase $09
CPI 0B

```



	JZ	CH400	10 tentokrát \$0B	MUS2:	LXI	H, 0FA	
	STA	PRECH	ani jeden z nich		SHLD	KLOPA	
	MVI	A, 1	vrát, to do bufferu		SHLD	KLOPB	
	STA	BUFINT			MVI	A, 1	
	CALL	PARAM	cti číslo		STA	CKLO	
	MOV	A, L			MVI	A, 30	
	STA	PRST	nastav počet r. na stránku		STA	KMIT	
	CALL	BUFE	odelovac pryc		JMP	MUSIC	
CH00:	JMP	STRAN	nastav novou stranu	MUS3:	LXI	H, 1F4	
	CALL	CR	udelej CR		SHLD	KLOPB	
	LXI	H, 2	nastav velikost znaku.		SHLD	KLOPA	
	SHLD	SIXX			MVI	A, 1	
	SHLD	SIYY			STA	CKLO	
	MVI	A, 78	a počet znaku		MVI	A, 70	
	STA	PZRA			STA	KMIT	
	RET				JMP	MUSIC	
CH400:	CALL	CR	udelej CR	MUSIC:	LDA	CKLO	ulastni generovani zvuku
	LXI	H, 3	nastav velikost		STA	CKLO1	
	SHLD	SIXX			LHLD	KLOPA	
	SHLD	SIYY			SHLD	KLOP1	
	MVI	A, 50	a počet znaku	MUSY:	CALL	BEEP	jedno pisknutí
	STA	PZRA			LHLD	KLOP1	
	RET				DCX	H	
CH26:	CALL	CR	udelej CR		MOV	A, L	
	LXI	H, 6	nastav velikost znaku		ORA	H	
	SHLD	SIXX			SHLD	KLOP1	
	SHLD	SIYY			JNZ	MUSY	odpiskano ? -> ne
	MVI	A, 28	a počet znaku		LDA	CKLO1	ano
	STA	PZRA			DCR	A	
	RET				STA	CKLO1	
PRINT:	LDA	RDSI	podle prepínací znak		JNZ	MUPRO	
	ANI	80			LHLD	KLOPB	
	CZ	PREVOD	proved	MUSN:	SHLD	KLOP2	
	MOV	A, C	neprovedej		LDA	KMIT	budeme chvíli potichu
	ANI	80	je-li s 8. bitem		CALL	WAITA	
	MOV	A, C			CALL	TLAC	testuj tlačítko
	JNZ	PTECK	tiskni tecku		LHLD	KLOP2	
	JMP	PRNI	není - tiskni jej		DCX	H	
PTECK:	MVI	A, 2E	kód tečky		MOV	A, L	
	JMP	PRNI	vytisknout		ORA	H	
NOVR:	PUSH	B	schovej si BC		JZ	MUSIC	
	CALL	CR	udelej CR		SHLD	KLOP2	
	CALL	LF	jeste LF		JMP	MUSN	
	POP	B	obnov BC	MUPRO:	MVI	B, 68	pauza
	RET		a zpět		CALL	P50	
PRNI:	CALL	PCHR	jdi tisknout		JMP	MUSIC+6	
	LDA	UKZR	počet znaku	TLAC:	LDA	RDSI	je-li stisk tl.
	INR	A	zvys o jednu		ANI	20	
	MOV	B, A			RNZ		ne
	LDA	PZRA	porovnej s maximálním	STISK:	LDA	RDSI	ano -> cekej
	CMP	B			ANI	20	na jeho pustení
	JZ	NOVR	stejně, udelej novou radku		JZ	STISK	
	MOV	A, B			POP	H	
	STA	UKZR	ulož to zpět		LDA	KR025	
	RET				ANI	7F	
PREVOD:	LXI	H, TABPR	adresa převodní tabulky		STA	RDS0	a zhasni diodu
	MOV	A, M	kód z tab. do A		RET		
	CMP	C	porovnej s převadeným	BEEP:	LDA	KLM	podle frekvence změni
	JZ	VYMEN	hura - je to on		XRI	20	stav reproduktoru
	CPI	0	není už konec tabulky ?		STA	KLM	
	RZ		je - navrát		ORI	80	+ rozsvícena dioda
	INX	H	není - další položka		MOV	B, A	
	INX	H			LDA	KR025	
	JMP	PREVOD+3	a znovu		ORA	B	
VYMEN:	INX	H	nasledující pozice		STA	RDS0	
	MOV	C, M	presun		LDA	KMIT	
	RET		a vrát se		CALL	WAITA	
STRAN:	XRA	A	znuluj		JMP	TLAC	a jeste otestuje tlačítko
	STA	UKZR	ukazatel znaku		DCR	A	smucka
	STA	UKRA	ukazatel radek		RZ		
	LDA	RDSI1	testuj typ papíru	CR:	JMP	WAITA	
	ANI	20			XRA	A	znuluj
	JZ	STRR	-> role		STA	UKZR	počet znaku
	CALL	MUS2	A4 = piskej		LXI	H, 0	
	LHLD	STPX	vypočti vzdálenost		SHLD	NOPX	tisk bude od kraje
	MVI	A, 1	od okraje		RET		
	STA	PRZ		LF:	LDA	UKRA	provede LF
	CALL	COMP	a násob -1		INR	A	
	XCHG				STA	UKRA	
	LXI	B, 190	konstanta Y pro nový zac.		LXI	H, PRST	
	CALL	MOVR	presun		CMP	M	test na počet radek
	JMP	NULXY	nuluj souřadnice		JZ	STRAN	maximální -> stránkuj
STRR:	CALL	MONOP	presun na pozici dalšího t.		CALL	VYPY	vypočti Y
	LXI	D, 0			MVI	A, 1	
	LXI	B, 0FF10	konstanta pro vynechání		STA	PRZ	
STRR1:	CALL	MOVR	mista		CALL	COMP	
	CALL	CR	udelej CR		XCHG		
	CALL	LF	jeste LF		LHLD	NOPY	
	XRA	A	znuluj ukazatel radek		DAD	D	
	STA	UKRA			SHLD	NOPY	tisk bude o radku níž
	RET		navrát		RET		
MUS1:	LXI	H, 64	inicializace zvukového	CHCOL:	LDA	COLR	nastavi následující barvu
	SHLD	KLOPA	generatoru na standardní		INR	A	
	LXI	H, 1	hodnotu pro 3 typy pískání		CPI	5	
	SHLD	KLOPB			JNZ	COL1	
	MVI	A, 1			MVI	A, 1	byla-li 4 nastavi znovu 1
	STA	CKLO			JMP	COL1	
	MVI	A, 30		MONOP:	LHLD	NOPX	presun na poz. příštího znaku
	STA	KMIT			XCHG		
	JMP	MUSIC			LHLD	NOPY	

```

MOV      B,H
MOV      C,L
JMP      MOVA
BASP:   LDA      UKZR      ;provede zpětný krok
DCR      A
INR      A
RZ       ;vrat se jsi-li na zač. řádky
DCR      A
STA      UKZR
CALL     VYPX
MVI      A,1
STA      PRZ
CALL     COMP
XCHG
LHLD     NOPX
DAD      D
SHLD     NOPX      ;X nastav na předchozí znak
RET
VYPX:   LHLD     SIXX      ;výpočet X podle
MOV      A,L        ;síry znaku
MVI      A,6
XCHG
LXI      H,0
JMP      NASOB
LINUP:  LDA      UKRA      ;provede posun o řádek zpět
DCR      A
INR      A
RZ       ;zač. stránky - vrat se
DCR      A
STA      UKRA
CALL     VYPY
XCHG
LHLD     NOPY
DAD      D
SHLD     NOPY      ;Y nastav o řádek zpět
RET
VYPY:   LHLD     SIYY      ;výpočet Y podle
MOV      A,L        ;výšky řádky
ANI      0FE
MOV      E,A
MVI      D,0
DAD      D
XCHG
LXI      H,0
MVI      A,8
CALL     NASOB
RET
PCR:    CALL     CR        ;podle prep. udělá CR
LDA      RDSI1
ANI      80
RNZ
JMP      LF
PLF:    CALL     LF        ;podle prep. udělá LF
LDA      RDSI1
ANI      40
RNZ
JMP      CR
FAST:   CALL     PARAM      ;nastav krok
MOV      A,L
ANI      1
RLC
RLC
RLC
RLC
STA      KR025
ANI      10
JNZ      F025
MVI      A,0
JMP      SETS+4
F025:   MVI      A,10
JMP      SETS+4
        ;prevod SHARP -> ASCII

TABPR:  DB      0D7,10,0FC,17,5E,18
DB      0F,0C,0E,9,0C,7
DB      0C6,19,5F,1A,0AE,1B
DB      0FB,1C,0FF,1E,0CF,1F
DB      0B,5E,93,60,0A1,61
DB      9A,62,9F,63,9C,64
DB      92,65,0AA,66,97,67
DB      98,68,0A6,69,0AF,6A
DB      0A9,6B,0B8,6C,0B3,6D
DB      0B0,6E,0B7,6F,9E,70
DB      0A0,71,9D,72,0A4,73
DB      96,74,0A5,75,0AB,76
DB      0A3,77,9B,78,0BD,79
DB      0A2,7A,0BE,7B,80,7D
DB      94,7E,0
        ;tabulka tisku pro test

TSTTB:  DB      2A,2A,2A,20,4D,5A
DB      2D,38,32,31,20,2A
DB      2A,2A

MOVA:   CALL     PNUP      ;zvedni pero
JMP      ABSO      ;presun na X,Y
VECTA:  CALL     PNDW      ;spust pero
JMP      ABSO      ;presun na X,Y
POINTA: CALL     PNUP      ;zvedni pero
CALL     ABSO      ;presun na X,Y
JMP      PNPT      ;udelej bod

```

```

MOV      B,H
MOV      C,L
JMP      MOVA
VECTR:  CALL     PNUP      ;zvedni pero
CALL     RELA      ;relativní presun
JMP      PNDW      ;spust pero
CALL     RELA      ;relativní presun
CALL     PNUP      ;zvedni pero
CALL     RELA      ;relativní presun
JMP      PNPT      ;udelej bod

PNPT:   CALL     PNDW      ;spust pero
JMP      PNUP      ;zvedni pero = bod

SPEED:  ADI      0
STA      SPED
RET

TEST:   JMP      VTEST

PORG:   MVI      A,14
STA      SPED
STA      P185
PUSH     D
PUSH     B
MVI      A,1
STA      SPEN
LXI      B,8
MOV      D,B
MOV      E,C
CALL     MOVR      ;par kroku doprava nahoru
MVI      A,00
STA      SPED
LXI      H,6A4
SHLD     STPX
LXI      H,0FE70
SHLD     STPY
POP      B
PUSH     B
LXI      D,0
CALL     MOVA      ;obnov Y - presun na zadane Y
POP      B
PUSH     B
LXI      D,50
CALL     MOVA      ;okamžite a na 0 do X
XRA      A
STA      STPX
STA      TPLI
CALL     NULXY
POP      B
POP      D
CALL     MOVA      ;obnov Y
MVI      A,0A
STA      P185
LXI      H,3
SHLD     SIXX
SHLD     SIYY
MVI      L,0
SHLD     SIXY
SHLD     SIYX
SHLD     ASCD
MVI      A,80
STA      NARW
STA      PSLI
RET
KTST:   MVI      A,0A
STA      P185
LXI      H,3
SHLD     SIXX
SHLD     SIYY
MVI      L,0
SHLD     SIXY
SHLD     SIYX
SHLD     ASCD
MVI      A,80
STA      NARW
STA      PSLI
RET
RELA:   LHLD     STPX
DAD      D
XCHG
LHLD     STPY
DAD      B
MOV      C,L
MOV      C,L
LHLD     STPX
XCHG
SHLD     STPX
SHLD     NOPX
CALL     SMR
PUSH     H
PUSH     B
MOV      C,A
POP      D
LHLD     STPY
XCHG
SHLD     STPY
SHLD     NOPY
CALL     SMR
RLC
RLC
RLC
RLC
MOV      B,A
ORA      C
DIRE
POP      D
RZ
MOV      A,L
SUB      E
MOV      A,H
SBB      D
MOV      A,C
DALX
XCHG
MOV      A,B
STA      DIR1
SHLD     DIFF
LXI      H,14

```

CALL	FMOT1		CALL	PCHR4	
MVI	A, 29		MOV	A, M	
STA	ACCE	; rozbehova rychlost	ORA	A	
MOV	B, D	; BC = rozdíl	JP	WRCON	
MOV	C, E		XRA	A	
XRA	A		STA	COUNZ	
SUB	E		CALL	PCHR3	
MOV	L, A		INX	H	; dalsi pozice znaku
MVI	A, 0		DCR	E	; zbyvajici pocet
SBB	D		JMP	WRIT1	; a znovu
MOV	H, A	; HL = DE * -1	H	H, BS	; neni-li nastaven bs
XRA	A		LXI	A	; na nulu nedela nic
MOV	A, D		CMP	M	
RAR	D, A		MOV	M, A	
MOV	A, E		JNZ	BSNA	
RAR	E, A	; DE = DE / 2	LHLD	NOPX	; jinak nastavi tisk znaku
ABS1:	PUSH	; schovej HL a DE	SHLD	PREX	; na novou pozici
PUSH	D		LHLD	NOPY	
XCHG		; DE <=> HL	SHLD	PREY	
MOV	H, B		POP	H	
MOV	L, C	; HL = BC	RET		
DAD	H		PCHR:	PUSH	H
DCX	H	; HL = HL * 2 - 1	LXI	H, BS	
DAD	D	; HL = HL + DE	CPI	8	
LXI	H, ACCE		JNZ	NOBS	
JNC	ABS2	; je-li HL >= 0 skoc	MOV	M, A	; ma-li kod \$08
LDA	P185		POP	H	; pouze jej uloz
CMP	M		RET		; a return
JZ	ABS2		ANI	7F	; pouze 7 bitu
ABS2:	DCR	M	MOV	C, A	
MVI	A, 28		CALL	BS08	; vyhodnot byl-li kod \$08
SUB	M		LDA	ASCD	
SUB	C		ORA	A	
MVI	A, 0		JNZ	NODIA	; skoc pri ASCII rezimu
SBB	B		MOV	A, C	
JC	ABS3		CPI	7B	
ABS3:	INR	M	JC	NODIA	; skoc neni-li to diakr. znak
LHLD	DIFF		ADI	5	; zvys na diakr. znamenska
POP	D		MOV	C, A	
DAD	D		INR	M	
XCHG			MOV	A, C	; znak do A
POP	H		CPI	10	; je-li mensi nez 10
PUSH	H		JC	PCHR2	; skoc
DAD	D		ANI	0F8	; ne - znuluj 3 nejnijsi bitu
LDA	DIR1		RAR		; vydel 2
JNC	ABS4		RAR		; 4
XCHG			ADI	0FC	; odedcti 3
LDA	DIRE		ADI	LOW ATB1	; + nizsi byte adresy tab.
ABS4:	PUSH	D	MOV	L, A	; do L
CALL	ELEM		MVI	A, 0	
DCX	B		ACI	HIGH ATB1	
MOV	A, B		MOV	H, A	; do H vyssi byte tabulky
ORA	C		MOV	A, M	
POP	D		INX	H	
POP	H		MOV	H, M	
JNZ	ABS1		MOV	L, A	; do HL adresu generatoru
SMR:	RET		MOV	A, C	
MOV	A, L	; vrati A = 0 pro skut. = zadane	ANI	7	; pouze 3 nejnijsi bitu
		; A = 1 pro skut. < zadane	MOV	C, A	
		; A = F pro skut. > zadane	JZ	VYZN	; jsou-li 0 skoc
SUB	E		MOV	A, M	; nejsou
MOV	L, A		ORA	A	
MOV	A, H		INX	H	
SBB	D		JP	SKZN	; testuj nejvyssi bit
MOV	H, A	; H = H - D - predchozi znam.	DCR	C	
DAD	H	; HL = HL * 2	JNZ	SKZN	; adresu na znak podle nich
JNC	SMVL	; -> je-li HL kladne	MVI	A, 8	; znak v CG vybrán
XRA	A	; zaporne - preved na kladne	CALL	PCHR4	
SUB	L		MOV	A, M	
MOV	L, A		CALL	PCHR4	
MVI	A, 0		MOV	A, M	
SBB	H		INX	H	
MOV	H, A	; HL = -1 * HL	ORA	A	
MVI	A, 0F	; priznak a navrat	JP	PCHR1	
RET			POP	H	
SMVL:	MOV	A, H	PCHR2:	POP	
ORA	L		PCHR3:	PUSH	H
RZ			LDA	NARW	
MVI	A, 1	; je-li HL = 0 vrat se	RLC		
RET		; jinak priznak a navrat	RAL		
WRITE:	XRA	A	ADI	0F6	
STA	COUNZ		CMA		
WRIT1:	MOV	A, E	MVI	C, 0	
ORA	A	; je-li delka nulova	CALL	PCHR5	
RZ		; vrat se	POP	H	
LDA	COUNZ		RET		
ORA	A	; neni-li 1. znak	MOV	C, A	
JNZ	WRIT3	; skoc	ANI	7	
MOV	A, M	; je to prvnii znak	PUSH	H	
ORA	A	; testuj je-li 0	PUSH	D	
JZ	WRIT2	; je - tak skoc	PUSH	PSW	
MOV	A, M	; neni, jdi jej vytisknout	LHLD	SIXX	
CALL	PCHR		XCHG		
JMP	WRCON		LHLD	PREX	
WRIT2:	INR	A	CALL	PCHR8	
STA	COUNZ		XCHG		
CALL	BS08		MOV	A, C	
WRIT3:	MOV	A, M	ANI	38	
			RAR		

RAR			MOV	A, H	
LXI	H, BS		ORA	L	
ADD	M		JNZ	FMOT1	
MOV	B, A		RET		
LHLD	SIYX				
XCHG			PRELI:	PUSH	B
CALL	PCHR8			PUSH	PSW
POP	PSW			ANI	80
PUSH	H			JZ	LINR
LHLD	SIYV			LDA	TPLI
XCHG				ORA	A
LHLD	PREY			JZ	LINR
CALL	PCHR8			LDA	AKLI
MOV	A, B			DCR	A
XCHG				JZ	LINZ
LHLD	SIYV			STA	AKLI
XCHG			LINR:	POP	PSW
CALL	PCHR8			MOV	B, A
PUSH	H			LDA	PSLI
MOV	A, C			ORI	7F
ANI	40			ANA	B
JZ	PCHR6			STA	MOTR
LHLD	NOPX			POP	B
XCHG				RET	
LHLD	NOPY		LINZ:	LDA	PSLI
MOV	B, H			ORI	7F
MOV	C, L			MOV	B, A
LDA	SPEN			POP	PSW
ORA	A			ANA	B
CZ	MOVA			STA	MOTR
POP	B			CALL	P50MS
POP	D			LDA	PSLI
CALL	VECTA			XRI	80
JMP	PCHR7			STA	PSLI
PCHR6:	CALL	PNUP		LDA	TPLI
	POP	H		STA	AKLI
	SHLD	NOPY		POP	B
	POP	H		RET	
	SHLD	NOPX			
PCHR7:	POP	D	P50MS:	MVI	B, 19
	POP	H	P50:	MVI	C, 0F4
	RET		P50A:	DCR	C
PCHR8:	INR	E		JNZ	P50A
	DCR	E		DCR	B
	RZ			JNZ	P50
	PUSH	H		RET	
	LXI	H, 0			
PCHR9:	DCR	A	DRZ:	MOV	L, A
	JM	PCHR0		ANI	0F
	DAD	D		JZ	DRZN
	JMP	PCHR9		CPI	1
PCHR0:	POP	D		LDA	RDSI
	DAD	D		JZ	DRZR
	RET			ANI	1
				JNZ	DRZN
ELEM:	CALL	DRZ	DRZY:	MOV	A, L
	LHLD	SMOT		ANI	0F0
	MOV	H, A		PUSH	PSW
	ADD	L		LDA	PPORG
	ANI	0F		ORA	A
	MOV	E, A		JNZ	HAVAR
	MOV	A, H		POP	PSW
	ANI	0F0		RET	
	ADD	L	DRZR:	ANI	2
	ANI	0F0		JZ	DRZY
	ORA	E	DRZN:	MOV	A, L
	STA	SMOT		RET	
	XRA	A			
	MOV	H, A	NULXY:	LXI	H, 0
	LDA	ACCE		SHLD	STPX
	RAR			SHLD	STPY
	MOV	L, A		SHLD	NOPX
	JMP	NASTM		SHLD	NOPY
	LDA	SPEN		RET	
	DCR	A			
	RNZ				
	JMP	NASTP			
PNDW:	LDA	SPEN			
	DCR	A			
	RZ				
	MVI	A, 1			
NASTP:	STA	SPEN			
	LXI	H, 1F4			
NASTM:	PUSH	H			
	LHLD	SPEN			
	MOV	A, H			
	ANI	0E			
	ORA	L			
	MOV	L, A			
	MOV	A, H			
	ANI	0E0			
	RRC				
	ORA	L			
	RRC				
	CALL	PRELI			
	POP	H			
FMOT1:	LDA	SPED			
FMOT2:	DCR	A			
	JNZ	FMOT2			
	DCX	H			

ulastni generator znaku

DB	64, 54, 4B, 0C8, 2C, 69, 60, 50
DB	49, 0CC, 3C, 4C, 49, 50, 60, 69
DB	0EC, 18, 5C, 64, 6B, 69, 60, 50
DB	49, 0CC, 0A, 72, 7B, 7C, 21, 0E3
DB	0, 43, 4C, 6C, 69, 60, 50, 51
DB	0D4
TB12:	DB 78, 28, 6B, 64, 0CC, 9, 4B, 4A
DB	6A, 69, 3A, 0FA, 1, 42, 4B, 6B
DB	6A, 3B, 0FB, 78, 2C, 58, 0CC, 9
DB	4B, 4A, 7A, 0F9, 68, 60, 69, 62
DB	4A, 22, 6B, 64, 0CC, 68, 60, 69
DB	6B, 64, 0CC, 09, 50, 60, 69, 6B
DB	64, 54, 4B, 0C9
TB13:	DB 0, 68, 6B, 64, 5C, 53, 0D0, 4
DB	6C, 69, 60, 58, 51, 0D4, 9, 69
DB	61, 6A, 6B, 0E4, 4B, 54, 5B, 59
DB	60, 69, 0EC, 0C, 4B, 52, 7A, 29
DB	0EB, 28, 50, 49, 4B, 54, 0EC, 28
DB	4A, 0EC, 28, 49, 62, 4B, 0EC
TB14:	DB 6C, 28, 0CC, 0, 43, 4C, 6C, 28
DB	58, 51, 0D4, 28, 6C, 4B, 0CC, 0B
DB	52, 5A, 61, 6A, 72, 0FB, 0A, 0FA
DB	9, 52, 5A, 63, 6A, 72, 0F9, 20
DB	69, 5B, 0E4, 39, 7B, 6B, 69, 79, 80
TB15:	DB 32, 0FB, 2A, 71, 7A, 73, 0EA, 39
DB	72, 0FB, 31, 7A, 0F3, 39, 79, 3B
DB	0FB
. DEPHASE	
. . . . .	
. PHASE RAM ; blok promennych	
. . . . .	
P160:	DS 2 ; velikost pisma XX
S1XX:	DS 2 ; velikost pisma XY
S1XY:	DS 2 ; velikost pisma YX
SIYX:	DS 2 ; velikost pisma YY
SIYY:	DS 2 ; velikost pisma YY
. . . . .	
STPX:	DS 2 ; okamzita poloha pera X
STPY:	DS 2 ; okamzita poloha pera Y
NOPX:	DS 2 ; poloha pristiho tisku zn. X
NOPY:	DS 2 ; poloha pristiho tisku zn. Y
. . . . .	
NARW:	DS 1 ; druh mezer mezi znaky
ASCD:	DS 1 ; rezim diakriticky/ASCII
. . . . .	
BS:	DS 1 ; citac znaku pri f. WRITE
COUNZ:	DS 1 ; citac znaku pri f. WRITE
. . . . .	
SPEN:	DS 1 ; okamzity stav pera
SMOT:	DS 1 ; stav krokovych motorku
. . . . .	
PREX:	DS 2 ; sour. predchoziho tisku X
PREY:	DS 2 ; " " " " " " " " Y
. . . . .	
SPED:	DS 1 ; aktualni rychlost posuvu
. . . . .	
ACCE:	DS 1 ; zrychlovani pri rozbehu
DIRE:	DS 1 ; smer do os Y a X
DIR1:	DS 1 ;
DIFF:	DS 2 ; rozdil sour. zadane a skut.
P182:	DS 1 ;
P183:	DS 1 ;
P184:	DS 1 ;
P185:	DS 2 ;
. . . . .	
AKLI:	DS 1 ;
PSLI:	DS 1 ;
TPLI:	DS 1 ;
. . . . .	
REZI:	DS 1 ; rezim textovy/graficky
. . . . .	
COLR:	DS 1 ; aktualni barva
. . . . .	
SPD:	DS 1 ;
RTLI:	DS 1 ;
. . . . .	
SKUTX:	DS 2 ; poloha 0 od poc. papiru u X
. . . . .	
PPORG:	DS 1 ; druh funkce dorazu
VOLB:	DS 1 ;
. . . . .	
; promenne pro rizeni zvuk.	
; generatoru	
KMIT:	DS 1 ; kmitocet piskani
KLOP1:	DS 2 ; pomocne
CKLO1:	DS 1 ; promenne
KLOP2:	DS 2 ; pro generator
KLOPA:	DS 2 ; delka pauzy mezi pisk.
KLOPB:	DS 2 ; delka pauzy mezi cykly
CKLO:	DS 1 ; pocet pisknuti
KLM:	DS 1 ; stav reproduktorku
. . . . .	
SIZE:	DS 1 ; velikost znaku pro GPRINT
. . . . .	
UKRA:	DS 1 ; ukazatel radek
UKZR:	DS 1 ; ukazatel znaku
. . . . .	
STWP:	DS 1 ;
PUYE:	DS 1 ;
PLUX:	DS 2 ;

PRST: DS 1	počet řádek na stránku	KR025: DS 1	krok 0.125 nebo 0.25
PZRA: DS 1	počet znaku na řádek	PRZ: DS 1	
PLNB: DS 1		CK: DB 0	pomocné proměnné pro řízení
POMB: DS 1		CW: DB 0	ukazovátka bufferu
PITCH: DS 2	rozteč dílku na osě	ZW: DB 0	
REPE: DS 1	počet dílku na osě	BUFF: DS 0FF	vlastní buffer 1/4 kbyte
PRECH: DS 1	čísť z bufferu nebo z BUFINT	DS 1	
BUFINT: DS 1	pomocné prodloužení bufferu	CGR: DS 1FF	prostor pro stack
		DS 30	
		STACK: END 0	

## Manuál pro plotter – tiskárnu

### Formát příkazů

Je 5 typů formátu příkazů, jak je popsáno dále.

1. Příkaz skládající se pouze z jednoho znaku (bez další specifikace) (A, H, I).
2. Příkaz skládající se ze znaku a jednoho dalšího parametru (L, C, S, Q).
3. Příkaz skládající se ze znaku a dvou parametrů. „:“ se používá k oddělení parametrů a kód CR na zakončení souboru parametrů.
4. Příkaz plus znakový řetězec (P).
5. Příkaz skládající se ze znaku a tří dalších parametrů (X).

„:“ se používá k oddělení parametrů.

### Zadání parametrů

1. Mezery, předřazené parametru jsou ignorovány.
2. Každé číslo, kterému je předřazeno znaménko „-“ je považováno za záporné.
3. Každý parametr je zakončen „:“ nebo kódem CR. Jestliže jako specifikaci parametru zadáme jiný znak, než číslo, všechny znaky následující tento znak jsou ignorovány.

### Zkrácené formáty zadání

1. Libovolný jednoznakový příkaz může být následován příkazem bez toho, abychom jej museli oddělovat kódem CR. Např. „HD100,200“ CR má stejnou hodnotu jako „H“ CR „D100,200“ CR.
2. Libovolný dvouznakový příkaz (znak + parametr) může být následován libovolným příkazem, musí však být odděleny „:“. Např. „L0,S1,Q0,C1D100,200“ CR je platný výraz.

### Změny hodnot vnitřních proměnných při změně režimu

Při přechodu z grafického do textového režimu dochází ke změnám hodnot vnitřních proměnných:

- > **Souřadnice X a Y.**  
Y-ová souřadnice je nastavena na 0 a začátek je přiřazen levé straně plochy, na kterou je možno psát.
- > **Směr tisku znaku.**  
Parametr Q je nastaven na 0.
- > **Velikost znaku**  
stupeň velikosti znaku je nastaven na 1 (80 zn. na řádek).
- > **Typ čáry.**  
Druh čáry je nastaven na 0 (plná čára).

### Velikost znaku

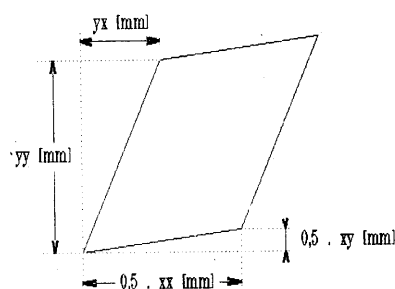
- > Po zapnutí tiskárny je velikost znaku automaticky nastavena na stupeň 1 (80 znaků na řádek). Stupeň velikosti je možno později měnit kontrolními kódy a příkazy.
- > V grafickém režimu je možno měnit velikost písma v rozsahu 0 až 63.
- > Při přepnutí z grafického do textového režimu je nastaven stupeň velikosti znaku na 1.

### Příkazy grafického režimu

Počítač řídí tiskárnu v grafickém režimu následujícími příkazy. Slova uvedená v závorkách, jsou příkazy v jazyce SHARP BASIC, které mají stejnou funkci.

Název příkazu	Formát a funkce
<b>Druh čáry</b>	<b>Lp</b> (p=0 až 15) Určuje druh čáry (plná nebo tečkovaná) a rozteč teček. p=0 platí pro plnou čáru, p=1 až p=15 pro různé tečkované čáry.
<b>Základní poloha (PHOME)</b>	<b>H</b> Zdvihne pero a vrátí jej do základní polohy.
<b>Inicializuj základní polohu (HSET)</b>	<b>I</b> Nastaví současnou polohu pera jako základní (x=0 a y=0).
<b>Kreslí (LINE)</b>	<b>Dx,y</b> (-9999<=x,y<=9999) Kreslí čáru ze současně polohy pera do bodu x, y.
<b>Kreslí (RLINE)</b>	<b>Jx,y</b> (-9999<=x,y<=9999) To samé jako předchozí příkaz, ale kreslí do bodu určeného relativními souřadnicemi x, y.
<b>Přesun (MOVE)</b>	<b>Mx,y</b> (-9999<=x,y<=9999) Zdvihne pero a přesune jej do bodu určeného souřadnicemi x, y.
<b>Přesun (RMQVE)</b>	<b>Rx,y</b> (-9999<=x,y<=9999) To samé jako předchozí příkaz, ale do bodu určeného relativními souřadnicemi x, y.

<b>Bod</b>	<b>Yx,y</b> (-9999<=x,y<=9999) Zdvihne pero a přemístí jej do bodu x,y a nakreslí bod.
<b>Bod</b>	<b>Wx,y</b> (-9999<=x,y<=9999) Totéž jako předchozí povel, ale bod udělá na místě určeném relativními souřadnicemi x, y.
<b>Změna barvy (PCOLOR)</b>	<b>Cn</b> (n=0 až n=3) Změní barvu pera podle čísla n. n=0 ... černá n=1 ... modrá n=2 ... zelená n=3 ... červená
<b>Nastav stupeň</b>	<b>Sn</b> (n=0 až 63) Určuje stupeň velikosti znaku.
<b>Otoč znak</b>	<b>Qn</b> (n=0 až 3) Určuje směr kterým se píše znaky.
<b>Tiskni</b>	<b>Pc1c2c3c4 ... cn</b> Tiskne znaky c1, c2 až cn
<b>Osa</b>	<b>Xp,q,r</b> (p=0 nebo 1; q=-999 až 999, r=1 až 255)  Je-li p=1 kreslí X-ovou osu, je-li p=0 kreslí osu Y-ovou, q určuje rozteč značek na ose a r jejich počet.
<b>Nastav velikost</b>	<b>Zxx,xy,yx,yy</b> Obdoba příkazu Sn + Qn, ale poskytuje značně větší možnosti. Nastaví směr, sklon a velikost písma.



XX ..... šířka písma v 1/2 | mm |,  
XY ..... odklon od x-ové osy v 1/2 | mm |,  
YX ..... odklon od y-ové osy v | mm |,  
YY ..... výška písma v | mm |.

Několik příkladů volby tisku příkazem „Z“:

XX,XY,YX,YY	tvar	výška	šířka	řádka
1,0,0,1	kolmý	1 mm	0.5 mm	vodorovně
4,0,0,7	stříhly kolmý	7 mm	2 mm	vodorovně
4,0,1,4	skloněn vpravo	4 mm	2 mm	vodorovně
0,6,-4,0	širší kolmý	4 mm	3 mm	svisle vzhůru
4,4,-4,4	kolmý	5.7 mm	2.3 mm	šikmo vzhůru

<b>Nastav rychlost</b>	<b>Tn</b> (n=-10 až 30) Nastaví rychlost tisku, n=-10 je největší a n=30 nejmenší rychlost.
<b>Nastav krok</b>	<b>Fn</b> (n=0 nebo 1) Nastaví krok 0.125 mm pro

n=0 nebo 0.25 mm pro n=1.  
Při kroku 0.125 mm se nastaví stupeň rychlosti na 0 a při kroku 0.25 mm je nastavován stupeň 16, což je třeba vzít v úvahu při nastavování rychlosti příkazem „T“.

**Nastav diakritický režim** Bn (n=0 nebo 1)

Nastaví diakritický režim při n=0. Je-li n=1 nastaví režim ASCII. Při tisku diakritických znamének nedochází k přesunu na tisk následujícího znaku, ale další znak se píše pod znaménko.

Tabulka odpovídajících znaků:

ASCII režim diakr. režim

{ 5Bh čárka  
| 5Ch kroužek  
} 5Dh háček

**Nadefinuj vlastní znak** En,c1c2c3...c32 (n=1 až 15)

Za znakem „E“ následuje číslo znaku n z rozsahu 1 až 15 a poté následují kódy podle tabulky uvedené v příloze tohoto manuálu. Může jich být maximálně 32, je-li jich méně, musí končit nastaveným 8.bitem.

Pro definici takového znaku je nejprve nutné nakreslit základní tvar (složený z úseček) do rastru 5x8 bodů, jednotlivým vrcholům (tj. koncům úseček) přiřadit čísla A1,A2,...,AN Tab. 2.

Horní čísla znamenají posun s písmem nahore, dolní posun s písmem dole. Poslední člen řetězce, který zajistí zvednutí písátka a návaznost pro psaní dalších znaků, musí mít 8.bit nastaven na 1 (tj. 80H). Dále lze ještě ukončovací člen (80H sloučit s posledním vrcholem kreslené úsečky (tj. (An+80H)).

Tabulka 2.

dec	hex				
56	38	57	39	58	3A
59	3B	60	3C		
120	78	121	79	122	7A
123	7B	124	7C		
48	30	49	31	50	32
51	33	52	34		
112	70	113	71	114	72
115	73	116	74		
40	28	41	29	42	2A
43	2B	44	2C		
104	68	105	69	106	6A
107	6B	108	6C		
32	20	33	21	34	22
35	23	36	24		
96	60	97	61	98	62
99	63	100	64		
24	18	25	19	26	1A
27	1B	28	1C		
88	58	89	59	90	5A
91	5B	92	5C		
16	10	17	11	18	12
19	13	20	14		
80	50	81	51	82	52
83	53	84	54		
8	8	9	9	10	0A
11	0B	12	0C		
72	48	73	49	74	4A
75	4B	76	4C		
0	0	1	1	2	2
3	3	4	4		
64	40	65	41	66	42
67	43	68	44		

Příklad vytvoření znaku „A“:

PRINT/P "E1,,"; CHR\$( \$08,\$7A,\$4C,\$21,\$E3)  
Vytiskni Gn (n=1 až 15)  
nadefinovaný znak Vytiskne dříve nadefinovaný znak čísla n z rozsahu 1 až 15.

## Kontrolní kódy používané v textovém režimu

Všechny kódy platí při přepínání ASCII/MZ800 nastaveném do polohy ASCII.

**Textový kód (\$01).**

Přepíná tiskárnu do textového režimu.

**Grafický kód (\$02).** (Jako příkaz PMODE v jazyku Basic)  
Přepíná tiskárnu do grafického režimu.

**Řádek zpět (\$03).** (Jako příkaz PSKIP v jazyku Basic)

Posune papír o 1 řádku zpět. Obsah čítače se přitom zmenší o 1. Stojí-li pero na začátku nové stránky, neprovádí nic.

**Zkouška per (\$04).** (Jako příkaz PTEST v jazyku Basic)

Nejprve vytiskne následující text, potom nastaví rozměr tisku 1 (80 znaků na řádek).

\*\*\* MZ-821 \*\*\*

**Tisk předdefinovaného znaku (\$05)+(\$02)+(\$01)+(\$0D).**

Vytiskne předdefinovaný znak čísla složeného z (ASCII)2 a (ASCII)1 z rozsahu 1 až 15.

**Zmenšený tisk (\$09)+(\$09)+(\$09).**  
Zmenšuje tisk ze stupně 1 na stupeň 0 (120 zn. na řádek).

**Zrušení zmenšeného tisku (\$09)+(\$09)+(\$08).**

Zvětšuje tisk ze stupně 0 na stupeň 1 (80 zn. na řádek).

**Nastavení čítače řádek.** (Jako příkaz PAGE v jazyku BASIC).

(\$09)+(\$09)+(ASCII)2+(ASCII)1+(ASCII)0+(\$0D)  
Určuje počet řádek na stránku tříbajtovým kontrolním kódem ASCII. Největší počet je 255. Po zapnutí nebo resetování systému je čítač řádek nastaven na hodnotu 66.

**Řádek vpřed (\$0A).** (Jako příkaz PSKIP v jazyku BASIC).

Posune papír o jeden řádek vpřed. Obsah čítače řádků se zvětší o 1.

**Zvětšený tisk (\$0B).**

Zvětšuje tisk ze stupně 1 na stupeň 2 (40 zn. za řádek).

**Zrušení zvětšeného tisku (\$07).**

Zmenšuje tisk ze stupně 2 na stupeň 1 (80 zn. na řádek).

**Návrat vozíku (\$0D).**

Přesune vozík (písátka) do levé krajní polohy.

**Krok zpět (\$08).**

Přesune vozík o jednu pozici zpět. Je-li v levé krajní poloze, kód nemá žádnou funkci.

**Nová strana (\$0C).**

Příkaz přesune papír na začátek další strany a zároveň vynuluje čítač řádek.

**Následující barva (\$1D).**

Změní barvu pera na následující podle tabulky uvedené výše.

Při výše uvedeném přepínání nastaveném do polohy MZ800 platí kódy podle následující tabulky:

Nová strana \$0F  
Zpětný krok \$0E  
Zrušení zvětšeného tisku \$0C

## Automatický test

Tiskárna umí 2 druhy testu. První se spouští řídicím kódem (\$04), nebo tlačítkem TEST na ovládacím panelu. Druhy spustíme stiskem tlačítka TEST a současným zapnutím tiskárny.

## Ovládací prvky na čelní stěně tiskárny

**Tlačítko STOP**

—slouží k pozastavení práce plotteru, tiskárna vydává tón a čeká na stisk tlačítka READY.

**Tlačítko TEST**

—provede to samé jako řídicí kód \$04, tedy test tiskárny.

**Tlačítko READY**

—při jakémkoli zvukovém signálu potvrzuje tímto tlačítkem uživatel prováděnou operaci.

**Přepínač PEN/NOT**

—slouží k invertování okamžitého stavu pera, provádí se technicky nezávisle na činnosti plotteru.

**Přepínač ASCII/MZ800**

—určuje mají-li se znaky přepočítávat (režim MZ800) či ne.

**Přepínač COLOR/NO**

—při příchodu instrukce pro změnu barvy a nastaveném přepínači do polohy COLOR vydává pískání odpovídající požadované barvě a čeká na stisk tlačítka READY.

**Dioda RDY**

—svitem určuje čekání na stisk tlačítka READY.

**Dioda PWR**

—signalizuje zapnutí plotteru.

**Dioda BUF**

—signalizuje vyčítání z bufferu, nebo svítí-li trvale, určuje prázdný buffer.

## Systémové přepínače na zadní stěně

**Přepínač CR/CR+LF** —v poloze CR+LF při příchodu kódu CR provede i kód LF.

**Přepínač LF/LF+CR** —v poloze LF+CR při příchodu kódu LF provede i kód CR.

**Přepínač A4/ROLE** —určuje formát papíru.

**Přepínač RDP/RDP** —určuje polaritu signálu RDP (hardwarově).

**Přepínač RDA/RDA** —určuje polaritu signálu RDA (hardwarově).

# EMULÁTOR TERMINÁLŮ CM7202/CM7209 EMU89

Ing. Petr Kandera, Krasnoarmějců 2, 704 00 Ostrava 3

EMU89 je program pro emulaci terminálů CM7202 a CM7209 na počítači typu IBM PC pod operačním systémem MS-DOS. IBM PC se připojuje jako terminál po sériové lince (RS232C, proudová smyčka).

Po spuštění EMU89 se zobrazí hlavní menu:

1. Emulace terminálu CM7202.
2. Emulace terminálu CM7209.
3. Změna parametrů přenosu.

Volit z těchto možností lze pomocí šipek a klávesy ENTER nebo přímo číslem volby. Kromě těchto možností je možné zobrazit popis funkčních kláves pro oba typy emulovaných terminálů stiskem klávesy K. Pokud do asi 5 sekund nestiskneme žádnou klávesu, automaticky se nastartuje „vysvícená“ volba. (Pozn.: časový údaj platí pro CPU 8088 a takt 9,54 MHz).

Pro svou činnost vyžaduje EMU89 v adresáři, ze kterého byl spuštěn, konfigurační soubor EMU89.CFG, ve kterém má uloženy parametry přenosu (číslo sériového portu, přenosovou rychlost, počet datových bitů, počet stop bitů, typ parity), číslo volby, která se po spuštění automaticky aktivuje, a označení diskové jednotky, na které bude hledán systémový soubor COMMAND.COM při práci s operačním systémem v emulátoru.

Všechny parametry, s výjimkou poslední, má uživatel možnost měnit volbou z pull-down menu. Poslední parametr se při instalaci vždy nastaví na C (harddisk). Konfigurační soubor lze měnit také libovolným textovým editorem.

Pokud v okamžiku startu neexistoval konfigurační soubor EMU89.CFG, automaticky bude nastartována volba 3 (Změna parametrů přenosu) a hlavní menu bude zobrazeno až po vytvoření konfiguračního souboru.

Program se ukončuje z hlavního menu pomocí CTRL-A.

## Změna parametrů přenosu

Parametry se mění výběrem z menu pomocí šipek a klávesy Enter, jak je vysvětleno v dolní části obrazovky. Klávesa Esc způsobuje vždy návrat o krok zpět bez změny hodnoty. Změnu parametrů lze ukončit stiskem Esc, pak se změny neuloží a platí pouze do ukončení programu, nebo stiskem CTRL-A – změny budou navíc uloženy do souboru. Pokud však před spuštěním neexistoval konfigurační soubor, i ukončení změn parametrů stiskem Esc způsobí uložení parametrů.

## Emulace

Při emulaci se chová PC jako zvolený typ terminálu s těmito funkčními klávesami:

- |           |  |
|-----------|--|
| <b>F3</b> | Lokální výmaz obrazovky.   |
| <b>F4</b> | Popis funkčních kláves (Help). Tato funkce odpovídá stisku klávesy K v hlavním menu. |
| <b>F9</b> | Změna parametrů přeno-   |

su. Tato funkce je ekvivalentní volbě 3 v hlavním menu.

**F10** Odskok do prostředí MS-DOS. Návrat zpět je možný zadáním sekvence EXIT. Pokud nebude nalezen COMMAND.COM na zvoleném disku, dojde po chybovém hlášení k návratu do emulace.

**CTRL-A** Ukončení emulace a návrat do hlavního menu.

**Enter** CR – „Návrat vozu“ (znak 13D).

**End** LF – skok na nový řádek (znak 10D).

**Delete** Delete – výmaz znaku.

**Back space** Back space (znak 08D).

Pro emulátor CM 7202 dále platí:

**F1** EOL – Výmaz řádku od kurzoru doprava.

**F2** EOS – Výmaz obrazovky od kurzoru dolů.

**Home** Home – Přesun kurzoru do levého horního rohu obrazovky.

\*(Pozn.: tyto funkční klávesy mají při emulaci CM7209 stejný význam jako neoznačené tři klávesy terminálu.)

Zbývající klávesy PC mají stejný význam jako klávesy terminálů nebo nemají žádnou funkci.

## Nároky na technické vybavení

Program EMU89 je určen pro počítače standardu IBM PC, pracující pod operačním systémem MS-DOS. Program byl zkoušen na počítačích Commodore PC20-III, PP06, Leonard AT, IBM PC-XT a IBM PS/2-30 s operačními systémy MS-DOS V3.20, V3.21, V3.30 a PPDOS V3.0. Vyžaduje grafickou kartu CGA nebo EGA (pro kartu Hercules stačí upravit atributy Text color z Text Background); v .EXE formě je dlouhý 22784 bajtů a v paměti zabírá asi 45 kB. Počítač musí být vybaven standardním sériovým portem COM1 nebo COM2.

## Stručný popis programu

Program je napsán v jazyce TURBO-PASCAL V4.0 (délka zdrojového programu je 520 řádků).

Program EMU89 se skládá ze čtyř základních částí:

1. pomocné globální procedury a funkce,
2. vlastní emulace,
3. procedury a funkce pro změnu parametrů,
4. hlavní program.

Hlavní program zajišťuje tvorbu hlavního menu a provádění základních operací s ním.

Používá mimo jiné proceduru Keys pro zobrazení popisu funkčních kláves (Help).

Změnu parametrů výběrem z pull-down menu a vytváření konfiguračního souboru má na starosti procedura

## Init,

hlavní procedura změny parametrů. Provádí výpis nastavených hodnot, volbu parametrů atd. Využívá procedury: Default (zobrazení nastavených hodnot), Save (ukládání parametrů do souboru), Mainwindow (okno pro změnu parametrů), Pspdf a Speed (volby jednotlivých parametrů). Dále používá procedury společné i pro hlavní program: Bottom (nápověď) a Arrows (pohyb kurzoru).

Emulace využívá tři hlavních procedur:

**Emulace** – Nastavení parametrů a volba typu emulátoru,

**Emu1** – emulace CM7202,

**Emu2** – emulace CM7209

a další pomocné procedury a funkce. Zjednodušeně lze popsat její činnost následovně:

Při spuštění emulace se nastaví parametry sériového portu a vektor přerušení. Přerušení při příjmu znaku na sériový port je obsluhováno procedurou Readcom, která ukládá znaky do pole, ze kterého jsou po zpracování znaky zobrazeny na obrazovce. Mezitím se soustavně testuje klávesnice – běžné znaky jsou vysílány přímo na port, funkční klávesy provádějí popisné funkce. Na obrazovce se zobrazují pouze znaky přicházející na sériový port.

Pomocné globální procedury a funkce zajišťují mimo jiné výstup znaku na port (OutChar), úschovu a obnovení obrazovky (SaveScreen, LoadScreen), odskok do MS-DOSu (Branch), inicializaci portu (Initcom), povolení a zablokování přerušení (SioInt On, SioInt Off), výmaz obrazovky od kurzoru dolů a doprava (ClrEos), přečtení znaku z pole s informací, zda se jedná o nový znak (InByte), zobrazení popisu funkčních kláves, změnu parametrů přenosu (Initid) apod.

Program lze snadno přizpůsobit individuálním požadavkům a vytvořit v něm vlastní speciální funkce, např. hardcopy obrazovky, nebo přiřadit dalším funkčním klávesám nové funkce.

Práce s emulátorem je jednoduchá a pohodlná, zvláště když uvážíme, že při práci s ním lze využít celou škálu programů přístupných pod MS-DOSem – v emulátoru lze provést odskok do MS-DOSu, pak v něm libovolně dlouho pracovat a kdykoliv se vrátit zpět do emulátoru.



```

{Emulator terminalu CM7202/CM7209}
{      na IBM PC      }
{      Verze 3.4      }
{      (c) 1989 - PKSoft      }

program EMU89;
{$K-U-C-}
{$M 20000,0,4000}
uses Crt,Dos;
label 5;
const
  sp='';comu1=$3F8;comu2=$2F8;
  EOI:byte=$20;BuffTop=10000;
  ter1='C M 7 2 0 2';ter2='C M 7 2 0 8';
  IntContr:integer=$20;q=$219;u=$220;
  IRQ_Mask:integer=$21;h=$223;
type
  strarr=array[1..8] of string[30];
var
  f:text;j:string[30];newbyte:boolean;
  term:string[12];r,z,v:string[2];
  keyout,c,drv:char;ScrSeg:word;
  p,x,d,b,s,t,m,e:strarr;
  ByteIn,ComInt,xx,yy,zac:byte;
  altcom,exitsave,screen:pointer;
  Inbuff:array[0..BuffTop] of byte;
  com,comnr,bd,db,sb,par,yl,yk:word;
  Origmode,Inpoint,Outpoint,i:word;

procedure ReadCom;
  Interrupt;
begin inc(Inpoint);
  if(Inpoint>BuffTop) then Inpoint:=0;
  Inbuff[Inpoint]:=port[com];
  port[IntContr]:=EOI;
end;

function Inbyte(var pb:boolean):byte;
begin pb:=(Outpoint<>Inpoint);
  if pb then begin inc(Outpoint);
    if Outpoint>BuffTop then Outpoint:=0;
    InByte:=Inbuff[Outpoint];end;
end;

procedure OutChar(z:char);
begin while(port[com+5] and $20 =0) do;
  port[com]:=byte(z);
end;

procedure SioInt_off;
begin
  Port[IRQ_Mask]:=(Port[IRQ_Mask] or $18);
end;

procedure SioInt_on;
begin
  Port[IRQ_Mask]:=
    (Port[IRQ_Mask] and $E7);
end;

procedure Initcom(comport:integer;
  baud:real;data,stop,parit:byte);
var divisor:integer;local:byte;
begin divisor:=round(115200.0/baud);
  port[comport+3]:=$80;
  port[comport]:= lo(divisor);
  port[comport+1]:=hi(divisor);
  local:=0;case data of 8:local:=3;
    7:local:=2;6:local:=1;end;
  if stop=2 then local:=local or 4;
  case parit of 2:local:=local or $18;
    1:local:=local or $08;end;
  port[comport+3]:=local;
  port[comport+1]:=1;port[comport+4]:=$0B;
  local:=port[comport];
end;

```

```

procedure ClrEos;
var i:byte;
begin ClrEol;xx:=wherex;yy:=wherey;
  for i:=yy+1 to 24 do begin
    gotoxy(1,i);ClrEol;end;
  gotoxy(xx,yy);
end;

procedure Title;
begin Textbackground(0);
  window(1,1,80,25);clrscr;
  Textcolor(0);Textbackground(7);
  Inline;gotoxy(16,1);writeln
    ('E M U L A C E   T E R M I N A L U   ',
  term);Textcolor(14);Textbackground(0);
  window(1,2,80,25);
end;

{$F+}procedure Myexit {F-};
begin Textmode(Origmode);SioInt_off;
  ByteIn:=port[com];ExitProc:=Exitsave;
  setintvec(comInt,altCom);
end;

procedure SaveScreen;
begin if (Lo(LastMode)=7) then
  ScrSeg:=$B000 else ScrSeg:=$B800;
  Move(Ptr(ScrSeg,0)^,Screen^,4000);
end;

procedure LoadScreen;
begin if (Lo(LastMode)=7) then
  ScrSeg:=$B000 else ScrSeg:=$B800;
  Move(Screen^,Ptr(ScrSeg,0)^,4000);
end;

procedure Init(u:byte);forward;
procedure Keys;forward;

procedure Initia(u:byte);
begin GetMem(Screen,4000);xx:=wherex;
  yy:=wherey;SaveScreen;
  if u=0 then Init(1)
    else Keys;LoadScreen;
  FreeMem(Screen,4000);Textcolor(14);
  Textbackground(0);window(1,2,80,25);
  gotoxy(xx,yy);
end;

procedure Branch;
begin GetMem(Screen,4000);xx:=wherex;
  yy:=wherey;SaveScreen;
  Textbackground(blue);window(1,1,80,25);
  clrscr;writeln('Navrat do emulatoru',
    -> <EXIT>');
  j:=drv+'\\COMMAND.COM';Exec(j,'');
  if DosError<>0 then begin writeln(#10,
    'Nenalezen soubor COMMAND.COM');
  write(#7);Delay(2000);end;
  Initcom(com,bd,db,sb,par);
  SetIntvec(ComInt,@readcom);
  SioInt_on;LoadScreen;
  FreeMem(Screen,4000);Textcolor(14);
  Textbackground(0);window(1,2,80,25);
  gotoxy(xx,yy);
end;

procedure Emul;
label 10;
begin while (keyout<>#1) do begin
  ByteIn:=InByte(newbyte);
  ByteIn:=(ByteIn and $7F);
  if newbyte then begin
    if i=2 then begin
      xx:=ByteIn-31;gotoxy(xx,yy);
      i:=0;goto 10;end;
    if i=1 then begin

```

```

yy:=ByteIn-31;Inc(i);
goto 10;end;
case ByteIn of 0:;7:write(#7);
8:gotoxy(wherex-1,wherey);
9:gotoxy(wherex+
(8-(wherex mod 8)),wherey);
11:gotoxy(wherex,wherey+1);
24:gotoxy(wherex+1,wherey);
26:gotoxy(wherex,wherey-1);
27:Inc(i);29:gotoxy(1,1);30:ClrEol;
31:ClrEos;else write(char(ByteIn));
end;end;
10:if keypressed then begin
keyout:=readkey;
if not (keyout=#1) then begin
if keyout=#0 then begin keyout:=readkey;
case ord(keyout) of 83: OutChar(#127);
79: OutChar(#10);72: OutChar(#26);
80: OutChar(#11);75: OutChar(#8);
77: OutChar(#24);71: OutChar(#29);
59: OutChar(#30);60: OutChar(#31);
61: clrscr;62: Initia(1);67: Initia(0);
68: Branch;15..25,30..38,46..50,63..66,
84..140,3,44,73,81,82;;end;end
else OutChar(keyout);end;end;end;
end;

```

```

procedure Emu2;
label 10;
begin while (keyout<>#1) do begin
ByteIn:=InByte(newbyte);
ByteIn:=(ByteIn and $7F);
if newbyte then begin
if i=3 then begin
xx:=ByteIn-31;gotoxy(xx,yy);
i:=0;goto 10;end;
if i=2 then begin
yy:=ByteIn-31;Inc(i);
goto 10;end;
if i=1 then begin Dec(i);
case ByteIn of
65:gotoxy(wherex,wherey-1);
66:gotoxy(wherex,wherey+1);
67:gotoxy(wherex+1,wherey);
68:gotoxy(wherex-1,wherey);
72:gotoxy(1,1);75:ClrEol;74:ClrEos;
89:i:=i+2;else;end;
goto 10;end;
case ByteIn of
0:;7:write(#7);9:gotoxy(wherex+
(8-(wherex mod 8)),wherey);
27:Inc(i);else write(char(ByteIn));
end;end;
10: if keypressed then begin
keyout:=readkey;
if not (keyout=#1) then begin
if keyout=#0 then begin keyout:=readkey;
case ord(keyout) of
83: OutChar(#127);79: OutChar(#10);
72: begin OutChar(#27);OutChar(#65);end;
80: begin OutChar(#27);OutChar(#66);end;
75: begin OutChar(#27);OutChar(#68);end;
77: begin OutChar(#27);OutChar(#67);end;
71: begin OutChar(#27);OutChar(#82);end;
59: begin OutChar(#27);OutChar(#81);end;
60: begin OutChar(#27);OutChar(#80);end;
61: clrscr;62: Initia(1);67: Initia(0);
68: Branch;15..25,30..38,46..50,63..66,
84..140,3,44,73,81,82;;end;end
else OutChar(keyout);end;end;end;
end;

```

```

procedure Emulace;
begin Title;i:=0; SioInt_off;
if comnr=1 then begin
com:=comu1;ComInt:=$0C;end
38 else begin com:=comu2;Comint:=$0B;end;

```

```

Initcom(com,bd,db,sb,par);
SetIntvec(ComInt,@readcom);
Inpoint:=0;Outpoint:=0;
SioInt_on;keyout:=$FF;OutChar(#13);
if term=ter1 then Emu1 else Emu2;
end;

```

```

procedure Arrows(ms:strarr;
ds,l,o:integer);
procedure Move(i:integer);
begin write(ms[wherey]);
gotoxy(wherex-ds,wherey+i);
if l=0 then Textbackground(0)
else Textbackground(green);
write(ms[wherey]);
gotoxy(wherex-ds,wherey);
if l=0 then Textbackground(white)
else Textbackground(0);
end;
begin c:=readkey;case ord(c) of
72: if wherey=1 then Move(o)
else Move(-1);
80: if wherey=o+1 then Move(-o)
else Move(1);
else write(#7);end;
end;

```

```

procedure Keys;
begin Textbackground(white);
window(1,1,80,25);clrscr;Textcolor(14);
gotoxy(7,24);write('Zbyvajici klavesy',
' PC odpovidaji terminalu nebo nemaji',
' zadny vyznam');gotoxy(30,22);
write('Navrat zpet -> <Esc>');
gotoxy(24,2);write('POPIS FUNKCNICH',
'KLAVES EMULATORU');Textbackground(0);
window(10,4,35,20);clrscr;
Textbackground(green);
write(' T E R M I N A L CM7202 ');
write(' ');
write(' klavesa funkce ');
gotoxy(1,5);Textbackground(0);
write(' F1 EOL ');
write(' F2 EOS ');
write(' F3 Vymaz obrazovky ');
write(' F4 Popis klaves ');
write(' F9 Zmena parametru ');
write(' F10 Odskok do DOSu ');
write(' Enter CR ');
write(' End LF ');
write(' Home HOME ');
write(' Delete(Del) DEL ');
write(' Backspace BS ');
write(' Ctrl-A Ukonceni emulace ');
Textbackground(0);window(45,4,70,20);
clrscr;Textbackground(green);
write(' T E R M I N A L CM7209 ');
write(' ');
write(' klavesa funkce ');
gotoxy(1,8);Textbackground(0);
write(' Enter CR ');
write(' End LF ');
write(' Delete(Del) DEL ');
write(' Backspace BS ');
write(' F3 Vymaz obrazovky ');
write(' F4 Popis klaves ');
write(' F9 Zmena parametru ');
write(' F10 Odskok do DOSu ');
write(' Ctrl-A Ukonceni emulace ');
window(1,1,80,25);gotoxy(47,22);
repeat repeat until keypressed;
c:=readkey;if c<>#27 then write(#7);
until c=#27;
end;

```

```

procedure Bottom(u:byte);
begin Textbackground(0);Textcolor(14);

```

```

window(4,22,77,24);clrscr;gotoxy(2,1);
if u=0 then gotoxy(19,1);
write(' Volba moznosti : ');
Textcolor(10);
write(#24,#25,' + Enter');
Textcolor(14);if u=1 then begin
write(' nebo primo ');Textcolor(10);
write(' cislo ');Textcolor(14);
write(' volby');gotoxy(2,3);
write(' Popis funkcnich klaves : ');
Textcolor(10);write('<K> ');
Textcolor(14);
write('Ukonceni emulace : ');end
else begin gotoxy(3,3);
write(' Navrat bez ulozeni : ');
Textcolor(10);write(' Esc ');
Textcolor(14);write(sp:9,
'Navrat s ulozenim : ');end;
Textcolor(10);write('Ctrl-A ');
end;

```

```

procedure Init(u:byte);
label 2;
var ch:byte;f:text;
procedure Default;
begin Textbackground(0);Textcolor(14);
window(50,4,77,18);clrscr;
Textbackground(green);write(sp:28);
write(' N A S T A V E N E HODNOTY ');
write;write(sp:28);write;
writeln;writeln;Textbackground(0);
write(' Seriovy port : ');
Textcolor(10);write(p[comnr]);
Textcolor(14);writeln;
write(' Rychlost prenosu : ');
Textcolor(10);write(bd:4);Textcolor(14);
writeln;write(' Parita : ');
Textcolor(10);write(x[par]);
Textcolor(14);writeln;
write(' Pocet dat. bitu : ');
Textcolor(10);write(db:4);Textcolor(14);
writeln;write(' Pocet stop bitu : ');
Textcolor(10);write(sb:4);Textcolor(14);
writeln;write(' Prvni volba : ');
Textcolor(10);write((yk:4);
Textcolor(14);
end;

```

```

procedure Save;
begin assign(f,'EMU89.CFG');rewrite(f);
writeln(f,comnr);writeln(f,bd);
writeln(f,par);writeln(f,db);
writeln(f,sb);writeln(f,yk);
writeln(f,drv);close(f);
end;

```

```

procedure MainWindow(m:strarr;n,l:byte);
var i,li:byte;
begin li:=23;
if l=0 then begin li:=3;
Textbackground(0);window(4,4,40,18);
clrscr;Textbackground(green);
write(sp:38);write(' Z M E N A ',
'P A R A M E T R U ');write(sp:36);
end;window(4,9,40,(9+n));
Textbackground(green);gotoxy(li,1);
write(m[1]);gotoxy(li,wherey+1);
Textbackground(0);
for i:=2 to n+1 do begin write(m[i]);
gotoxy(li,wherey+1);end;gotoxy(li,1);
end;

```

```

procedure Pdspf
(m:strarr;ds,o:integer;var res:word);
begin MainWindow(m,o,1);
repeat repeat until keypressed;
c:=readkey;if c=#0 then Arrows(m,ds,1,o)

```

```

else begin case ord(c) of
27:exit;13:begin res:=wherey;exit;end;
else write(#7);end;end;
until false;
end;

```

```

procedure Speed;
begin MainWindow(e,7,1);
repeat repeat until keypressed;
c:=readkey;if c=#0 then Arrows(e,5,1,7)
else begin case ord(c) of
27:exit;13:begin case wherey of
1:bd:=9600;2:bd:=4800;3:bd:=2400;
4:bd:=1200;5:bd:= 600;6:bd:= 300;
7:bd:= 150;8:bd:= 110;else bd:=9600;
end;exit;end;else write(#7);end;end;
until false;
end;

```

```

begin ch:=1;if u=0 then begin
{$I-}assign(f,'EMU89.CFG');
reset(f);close(f);{$I+}
if IOResult=0 then begin
assign(f,'EMU89.CFG');reset(f);
readln(f,comnr);readln(f,bd);
readln(f,par);readln(f,db);
readln(f,sb);readln(f,yk);
readln(f,drv);close(f);ch:=0;end;end;
if ch=1 then begin
if u=0 then begin comnr:=1;
bd:=9600;par:=3;db:=8;sb:=1;yk:=1;
drv:='C';end;Textbackground(white);
window(1,1,80,25);clrscr;Textcolor(14);
gotoxy(25,2); write('Emulator ',
'ternalu E M U 8 9');Bottom(0);
2:Default;MainWindow(m,5,0);
repeat repeat until keypressed;
c:=readkey;if c=#0 then Arrows(m,18,1,5)
else begin case ord(c) of
27:begin if u=0 then Save;exit;end;
1 :begin Save;exit;end;
13:begin case wherey of
1:begin Pdspf(p,4,1,comnr);goto 2;end;
2:begin Speed;goto 2;end;
3:begin Pdspf(x,5,2,par);goto 2;end;
4:begin Pdspf(d,6,1,db);
db:=db+6;goto 2;end;
5:begin Pdspf(b,10,1,sb);goto 2;end;
6:begin Pdspf(t,13,2,yk);goto 2;end;
else begin write(#7);goto 2;end;end;
end; else write(#7);end;end;
until false;end;
end;

```

```

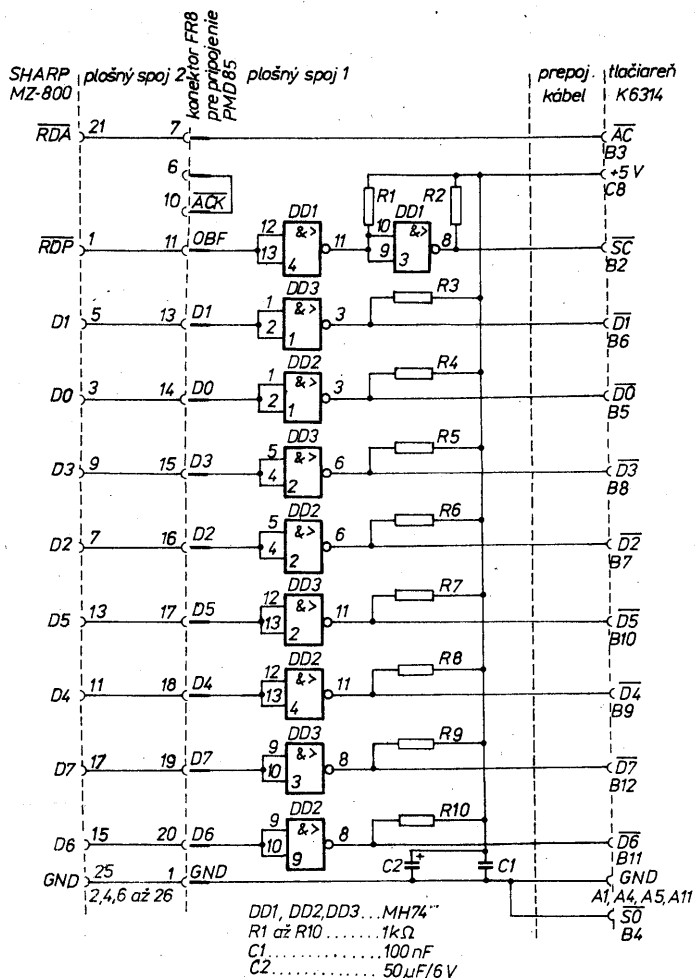
begin
r:=q+q;term:=ter1;z:=u+u;v:=h+h;zac:=0;
s[1]:='1. Emulace terminalu CM7202';
s[2]:='2. Emulace terminalu CM7209';
s[3]:='3. Zmena parametru prenosu';
m[1]:=' Seriovy port ';
m[2]:=' Rychlost prenosu ';
m[3]:=' Parita ';
m[4]:=' Pocet dat. bitu ';
m[5]:=' Pocet stop bitu ';
m[6]:=' Prvni volba ';
p[1]:='COM1';p[2]:='COM2';e[1]:=' 9600';
e[2]:=' 4800';e[3]:=' 2400';
e[4]:=' 1200';e[5]:=' 600';
e[6]:=' 300';e[7]:=' 150';
e[8]:=' 110';x[1]:=' licha';
x[2]:=' suda';x[3]:=' zadna';
d[1]:='7 bitu';d[2]:='8 bitu';
b[1]:='1 stopbit';b[2]:='2 stopbity';
t[1]:=' CM 7202';
t[2]:=' CM 7209';
t[3]:=' Inicializace';
GetIntVec(ComInt,altcom);

```

```

gotoxy(26,yl);write(s[wherey]);
Textbackground(white);case yl of
1:begin gotoxy(26,2);write(s[2]);
gotoxy(26,3);write(s[3]);gotoxy(26,1);
end;2:begin gotoxy(26,3);write(s[3]);
gotoxy(26,1);write(s[1]);gotoxy(26,2);
end;3:begin gotoxy(26,1);write(s[1]);
gotoxy(26,2);write(s[2]);gotoxy(26,3);
end;end;repeat i:=0;if zac=1 then begin
repeat Inc(i);until keypressed or
(i=15000);if i=15000 then c:=#13 else
c:=readkey;Inc(zac);end
else begin repeat until keypressed;
c:=readkey;end;
if c=#0 then Arrows(s,27,0,2)
else begin case ord(c) of
1: exit;13: begin case wherey of
2: begin yl:=2;term:=ter2;Emulace;end;
3: begin yl:=3;Init(1);end;
else begin yl:=1;term:=ter1;Emulace;end;
end;goto 5;end;
49: begin yl:=1;term:=ter1;Emulace;
goto 5;end;
50: begin yl:=2;term:=ter2;Emulace;
goto 5;end;
51: begin yl:=3;Init(1);goto 5;end;
107,75:begin yl:=wherey;Keys;goto 5;end;
end write(#7);end;end;until false;
end

```



Obr. 4. Schéma pripojenia (903-4)

DD1.3 a DD1.4 slúžia na výkonové prispôsobenie signálu OBFn z počítača PMD 85-2, ktorý je vedený priamo z MHB8255A bez výkonového zosilnenia. Vstupné obvody IRPR tlačiarne sú riešené odporovým prispôbením s rezistormi malých odporov

a priame pripojenie by mohlo viesť k poškodeniu MHB8255A. V prípade počítača Sharp MZ800 vlastne tieto hradlá nemajú význam, pretože všetky jeho výstupy sú výkonovo prispôbené. Signál ACn sa z tlačiarne privádza priamo na vývod č. 7 konektora

FRB, odkiaľ je v prípade počítača Sharp MZ800 vedený ako signál RDA priamo do počítača. V prípade PMD 85-2 treba tento signál invertovať. Na to sa využíva inverter, ktorý je na konektore FRB v PMD85-2 vyvedený. Vývod č. 7 je vstupom tohoto invertora, jeho výstup je na vývode č. 6, odkiaľ je signál vedený na vývod číslo 10 ako signál ACKn.

Signál SON je natrvalo pripojený na log.0.

Napájacie napätie +5 V pre integrované obvody, je použité z tlačiarne cez nevyužitý vývod č. C8.

Realizácia „redukcie“ pre pripojenie Sharp MZ800 spočíva iba v pripevnení vhodných konektorov na spoločnú dosku a ich vzájomnom prepjení drátovými spojmí. Prepaje sú jasné zo schémy (obr. 4).

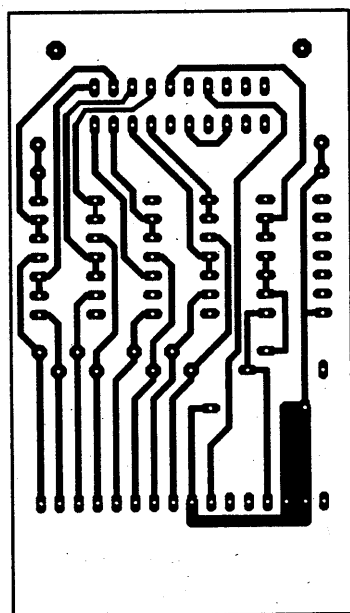
Oživenie a uvedenie do chodu by nemalo robiť problémy vzhľadom na jednoduchosť zapojenia. Nesmie sa však zabudnúť v interfejsu IRPR tlačiarne vyvieš na príslušný nevyužitý vývod (u nás C8) napájacie napätie +5 V).

## Zoznam súčiastok

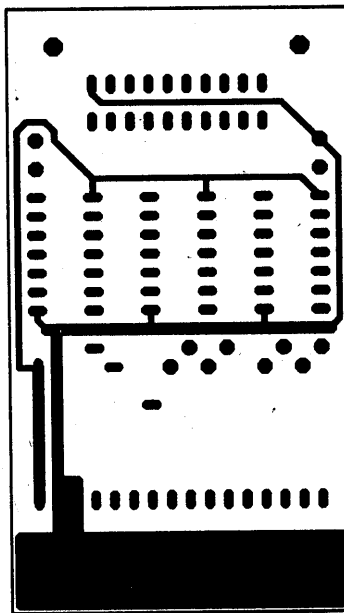
DD1, DD2, DD3	MH7438
R1 + R10	1kΩ, TR 211
C1	100nF, TK 783
C2	50 μF/6V, TE 981
konektor FRB	TY5122011/20
konektor FRB	TX5112013/20
pre Sharp MZ800	
konektor	WK46580 (skrátенý na 2×13 vývodov) pre Sharp MZ800

## Literatúra

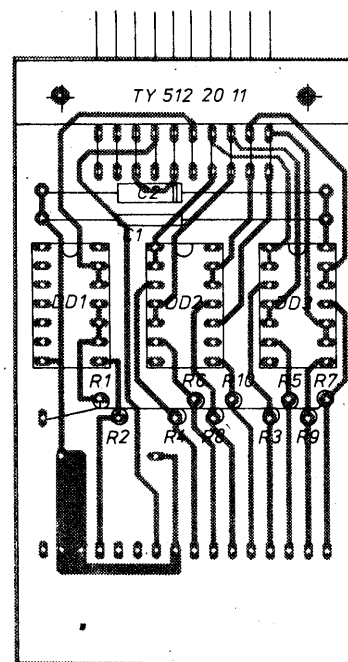
- Centronics. Amatérské rádio A 10/1984, str. 382.
- Systém malých elektronických počítačov SMEP, Abecednočíslicový displej SM 72-00. Technická dokumentácia, Vydavateľstvi Novinár Praha pre Teslu Orava.
- Technický popis modulu UKPR-8. Technická dokumentácia TESLA.
- MZ 800. Owner's manual, Sharp.
- PMD 85. Učebný materiál kurzu „Basic G“, Datasystem.
- Kočíš, I., Sulko, I.: Mikroprocesory a mikropočítače. ALFA, Bratislava – 1986.



Obr. 5. Obrázec plošných spojov zo strany spojov na doske Y504 (903-5)



Obr. 6. Obrázec plošných spojov zo strany súčiastok na doske Y504 (903-6)



Obr. 7. Rozmiestnenie súčiastok na doske Y504 (903-7)

# GENERÁTOR TISKOVÝCH SESTAV GTS

Ing. Ivo Křepinský, Karlovarská 5, 301 12 Plzeň

Generátor tiskových sestav slouží k ulehčení práce při vytváření programů pro tisk. Tvoří jej soubor procedur v Turbo Pascalu, které je nutno připojit k programu.

Tisková sestava se tiskne podle předlohy vytvořené předem textovým editorem. V předloze jsou vyznačena pole pro tisk hodnot. Do těchto polí se při běhu programu doplňují konkrétní údaje a výsledná sestava se tiskne.

Tiskovou sestavu je možno nejen vytisknout na tiskárně, ale i na jiném zařízení (terminál), případně ji uložit do souboru.

## Výhody GTS:

- jednoduchý a rychlý návrh tiskové sestavy,
- zkrácení programu pro tisk,
- snadnější ladění bez složitého vypočítávání polohy na papíru,
- oddělení grafické a obsahové složky tisku,
- možnost úprav tiskové sestavy bez zásahu do programu,
- navržená tisková sestava slouží zároveň jako část dokumentace,
- výstup je možno ve stejném tvaru posílat na libovolné výstupní zařízení nebo do souboru.

Generátor tiskových sestav je vytvořen v Turbo Pascalu v.3 a je použitelný v programech napsaných v tomto jazyku na všech počítačích, kde pracuje překladač tohoto jazyka.

## Vytvoření předlohy

Předloha pro tiskovou sestavu se vytváří libovolným textovým editorem (např. WordStar). Obsahuje pevné texty sestavy v tom tvaru, jak mají vypadat na výstupu. Dále obsahuje pole pro výstup hodnot. Tato pole jsou označena znakem "-" (podtržítka). Délka pole je určena počtem podtržítok.

Pole musí být na jedné řádce, nesmí být rozděleno na více řádek. Není možno umístit dvě pole těsně za sebou bez oddělení jiným znakem.

Soubor obsahující předlohu tiskové sestavy musí mít standardní příponu ".TS".

## Použití GTS

### Vyplňování hodnot

Program nejprve musí přečíst ze souboru předlohu tiskové sestavy (procedura TSREAD). Předloha setrvává v paměti a je možno ji používat, dokud není přečtena jiná.

Po přečtení předlohy je možno postupně vyplňovat pole pro hodnoty. Aktivní pole je určeno polohou ukazovátka. Po přečtení předlohy, po tisku sestavy a po užití procedury TSFIRST je ukazovátka nastaveno na první pole tiskové sestavy. Každý zápis hodnoty (procedura TSPRINT) posunuje ukazovátka na další pole. Procedura TSSKIP umožňuje přeskočit pole bez vyplnění.

### Ovládání výstupního souboru

Výstup je možno posílat do libovolného souboru, který je otevřen v Turbo Pascalu. Při použití standardního terminátoru (CON:), je možno výstup posílat i na výstupní zařízení.

Před všemi výstupy je nutno soubor otevřít (procedura TSOOPEN), po ukončení uzavřít (procedura TSCLOSE).

Všechny výstupy (procedury TSOUTPUT a TSWRITE) se posílají do otevřeného souboru.

### Použití generátoru v programu

Na počátku programu je nutno nadefinovat konstantu TSLEN, která udává maximální počet znaků největší předlohy, která bude v programu použita.

Generátor se k programu připojí direktivou `[$I GTS]`.

## Procedury pro práci s generátorem

### TSOPEN(Soubor:string| 14 |)

- otevírá výstupní soubor,
- tato procedura musí být použita před všemi výstupy (TSOUTPUT a TSWRITE),
- může být uvedeno libovolné jméno souboru (tiskárna je LST:, terminál CON:),
- všechny výstupy prováděné procedurami TSOUTPUT a TSWRITE se provádějí do otevřeného souboru.

### TSCLOSE

- zavírá výstupní soubor,
- je nutno použít, pokud je výstupním souborem diskový soubor, jinak se data nezapišou.

### TSREAD(Soubor:string| 14 |)

- čte předlohu tiskové sestavy ze zadaného souboru,
- jméno souboru se zadává bez přípony – ta je vždy .TS,
- ukazovátka se nastaví na první pole sestavy,
- předloha sestavy zůstává v paměti až do přečtení jiné předlohy.

### TSPRINT(Hodnota:string| 255 |)

- zapisuje hodnotu do tiskové sestavy,
- zápis se provádí vždy do současného pole,
- po zápisu se ukazovátka nastaví na následující pole,
- při zápisu se začíná od prvního znaku výstupního pole,
- znaky hodnoty přesahující délku pole se odříznou,
- zbývající znaky do délky pole se doplní mezerami,
- zápis se provádí pouze do paměti, výstup se provede až procedurou TSOUTPUT.

### TSSKIP

- přeskakuje současné pole,
- ukazovátka se nastaví na následující pole.

### TSFIRST

- nastavuje ukazovátka na první pole sestavy.

### TSOUTPUT

- zapiše kompletní sestavu do výstupního souboru,

- místo nevyplněných polí se tisknou mezery,
- po skončení zápisu se nastaví ukazovátka na první pole sestavy a je možno znovu vyplňovat pole.

### TSWIRE(Retezec:string| 255 |)

- zapiše řetězec do výstupního souboru,
- slouží pro nestandardní výstupy a pro ovládání tiskárny (např. odstránkování),
- řetězec se do výstupního souboru zapiše přesně v zadaném tvaru,
- zápis do výstupního souboru se provádí ihned.

*Poznámka:* Pokud mají procedury parametry, jsou všechny typu řetězec. Procedury nevracejí žádné hodnoty.

## Popis realizace GTS

Generátor tiskových sestav je navržen jako soustava procedur. Pro přehlednost a názornost jsou procedury dále členěny na menší podprogramy.

Názvy všech objektů (typů, proměnných a podprogramů), které jsou deklarovány jako globální, ale jsou určeny pouze pro GTS, začínají znaky "TS\_", aby byla vyloučena možnost kolize. Názvy všech objektů, které je možno v programu využívat, začínají znaky "TS".

### Pomocné proměnné

#### TS\_M:array| 0 .. TSLEN | of char

pole pro předlohu tiskové sestavy, naplňuje se při TSREAD; výstupní pole jsou označena nastavením nejvyššího bitu na 1.

#### TS\_Ptr:Integer

ukazatel do pole TS\_M na aktuální znak; obsahuje polohu, kde se bude číst nebo zapisovat.

#### TS\_Soubor:text

výstupní soubor.

### Pomocné procedury a funkce

#### TS\_Init

inicializace ukazatele TS\_Ptr.

#### TS\_Put (Znak:char)

zápis znaku do pole TS\_M a zvýšení ukazatele TS\_Ptr.

#### TS\_Get:char

čtení znaku z pole TS\_M a zvýšení ukazatele TS\_Ptr.

#### TS\_Dirget:char

čtení znaku z pole TS\_M bez zvýšení ukazatele TS\_Ptr.

**TS\_IsF(Znak:char):boolean**  
určuje, zda je daný znak součástí pole pro výstup hodnoty.

**TS\_SetF(Znak:char):char**  
vrací zadaný znak jako součást pole pro výstup hodnoty.

**TS\_GetF(Znak:char):char**  
vrací zadaný znak bez označení jako součást pole pro výstup hodnoty.

**Poznámka:**  
Realizace je navržena tak, aby bylo možno provést některé změny. Pomocí procedur a funkcí je oddělen problém uschování předlohy tiskové sestavy a problém označení polí pro výstup hodnoty.

## Popis algoritmu hlavních procedur

**TSOPEN**  
– přiřazení názvu souboru k TS.Soubor,  
– otevření pro zápis.

**TSCLOSE**  
– zavření souboru.

**TSPREAD** – lokální proměnné:  
Soubor:file – soubor předlohy.  
Sektor:array | 0 . . 127 | of char – oblast pro čtení sektoru.

- přiřazení názvu předlohy k Soubor a otevření pro čtení,
- inicializace TS\_Ptr,
- čtení sektoru ze Soubor až do konce souboru:
  - pro každý sektor,
  - Převod znaku ' ' na vnitřní označení výstupního pole,
  - uložení znaku do paměti,
- zavření souboru předlohy,
- přidání 'Z' na konec,
- nastavení prvního pole (TSFIRST).

**TSFIRST**  
– inicializace TS\_Ptr,  
– vyhledání prvního znaku s označením výstupního pole.

**TSSKIP**  
– přeskocení znaků s označením výstupního pole  
– vyhledání následujícího znaku s označením výstupního pole.

**TSPRINT**  
– od současné polohy se hodnota zapisuje až do délky pole,  
– zbývající znaky hodnoty se ignorují,  
– pokud není vyčerpána délka pole, doplní se mezerami,  
– skok na další pole (TSSKIP).

**TSOUTPUT**  
– inicializace TS\_Ptr,

- celá sestava, znak po znaku až do 'Z', se zapiše do souboru,
- nastaví se první pole (TSFIRST).

## Návrh úprav programu

**Zjednodušení programu:**  
– většinu pomocných podprogramů je možno zrušit a jejich činnost přenést na místo jejich volání – program se tím zkrátí,  
– výstup je možné omezit pouze na tiskárnu; pak je možno zrušit procedury TSOOPEN a TSCLOSE a zjednodušit výstupy.

**Vylepšení programu:**  
– přidělovat paměť pro předlohu tiskové sestavy dynamicky,  
– vytvořit zásobník předloh v paměti a tím umožnit opakované použití předlohy bez jejího nového čtení,  
– umožnit v proceduře TSSKIP přeskakování polí nazpátek a přeskakování o zadaný počet polí,  
– zavést různé formátování výstupu hodnot do polí sestavy (např. zarovnání vpravo, na daný počet desetinných míst) (většina těchto úprav však s sebou přináší značné zvětšení programu bez patrných výhod pro programátora).

Realizace v jiných jazycích:

Navržený algoritmus GTS je možno realizovat i v jiných programovacích jazycích téměř shodně.

Pro počítače, které nemají diskovou paměť, by bylo možné algoritmus upravit tak, aby pracoval s předlohou tiskové sestavy zadanou jako data v programu. Tím se však odstraňuje jedna z hlavních výhod GTS – jednoduché a názorné vytváření předlohy.

## Příklad použití

Použití GTS je předvedeno na jednoduchém příkladu. Program tiskne seznam pracovníků z diskového souboru.

Každý list sestavy se skládá z hlavičky a výpisu údajů o několika zaměstnancích. Program vytváří dvě předlohy pro tyto dvě části tiskové sestavy – jednu pro hlavičku a jednu pro údaje o jednom zaměstnanci. Na každou stranu se pak tiskne nejprve hlavička s doplněným datem a číslem strany a pak daný počet zaměstnanců (pro každého se doplní jeho pořadové číslo a další údaje ze souboru).

V příloze je uveden výpis programu, předloha pro hlavičku sestavy (obr. 1) a příklad sestavy (obr. 2).

## Výpis 1. Generátor tiskových sestav (930-V1)

```

*****
*                                     *
*                                     *
*          GTS                      *
*   generátor tiskových sestav      *
*                                     *
*   Ing. Ivo Krepinsky              *
*   (c) 1988, 1989                  *
*                                     *
*****

```

**Popis:**  
Soubor obsahuje procedury pro tisk sestav.

**Použití:**  
Soubor se v programu přidává direktivou \$I T6. Na začátku programu musí být provedeno nastavení konstanty TSLEN – maximální délka tiskové sestavy v bytech )

Strana: \_\_\_\_\_

Datum: \_\_\_\_\_

### SEZNAM PRACOVNIKU

Cislo: \_\_\_\_\_ Prijmeni: \_\_\_\_\_  
Jmeno: \_\_\_\_\_  
Datum narozeni: \_\_\_\_\_ Misto: \_\_\_\_\_

Obr. 1. Předloha obou částí sestavy (930-1)

Strana: 1

Datum: 01-08-89

### SEZNAM PRACOVNIKU

Cislo: 1 Prijmeni: Novak  
Jmeno: Josef  
Datum narozeni: 01-01-40 Misto: Plzen

Cislo: 2 Prijmeni: Soukup  
Jmeno: Jiri  
Datum narozeni: 10-02-56 Misto: Praha

Obr. 2. Příklad sestavy (930-2)

44

**Výpis 2. Ukázkový program použití GTS**  
(930-V2)

# HARDWAROVÁ NÁSOBIČKA MH102

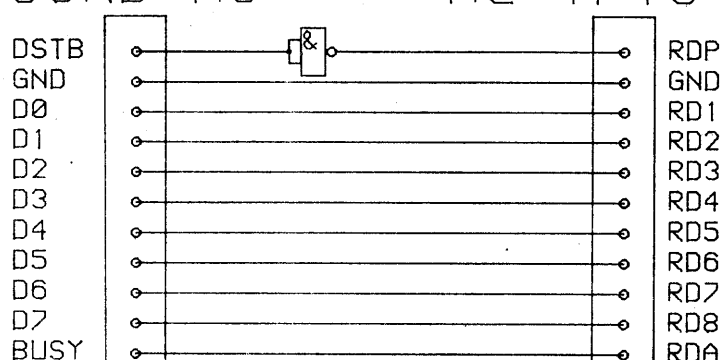
Článek se zabývá použitím hardwarové násobičky MH102 v osmibitových mikropočítačích. Při praktických aplikacích mikropočítačů např. v měřicí technice se dostává do popředí problém aritmetického zpracování naměřených dat. Čistě programové řešení výpočtu je obvykle časově náročné, a proto se k jeho urychlení používají aritmetické koprocesory (např. I 8231, I 8232 apod.), které se však obtížně shánějí a jsou drahé. Přitom v řadě aplikací vystačíme kromě součtu a rozdílu pouze s násobením, které ovlivňuje celkovou dobu výpočtu rozhodující měrou.

Principiální zapojení násobičky je na **obr. 1**. Obvod MH102 obsahuje 6 řídicích

Při praktické realizaci je nutné připojit obvod na datovou sběrnici mikropočítače přes







# EXPERTÍK

Ing. Jiří Mitlöhner, Zahradní 228, Sázava, PSČ 285 06

## Účel programu:

1. Příklad využití programovacího jazyka LOGO v oblasti zpracování seznamů a textů.
2. Předvádí a učí základní činnosti s databázovými systémy.
3. Umožňuje vytvořit jednoduché znalostní a výukové systémy.
4. Seznamuje s problematikou tvorby a využití jednoduché báze znalostí.

## Spuštění programu

Po vložení diskety zapišeme LOGO a tím zavedeme do operační paměti interpret jazyka LOGO.

Na obrazovce vypsané texty echa interpreta odpovídají

- a) 40 znakům v řádku obrazovky, pak zapišeme LOAD „EXPERT1“,
- b) 80 znakům v řádku obrazovky, pak zapišeme LOAD „EXPERT“.

Dále postupujeme podle popisu funkce programu a nápovědy na obrazovce.

## Popis báze znalostí

Báze znalostí je reprezentována stromovou datovou strukturou, kde každý uzel je označen kódem, (kód odstavce libovolná znaková kombinace), popsán textem (text odstavce o max. délce 128 znaků) a je spojen s jedním nebo více následnými uzly odkazem na jejich kód (odkaz). Každé spojení může být popsáno textem (text odkazu). Stromová struktura báze znalostí může být téměř libovolně větvená i se zpětným provázáním. Programově je každý uzel báze znalostí reprezentován složeným seznamem, kód uzlu je jménem seznamu. Kódy uzlu jsou uloženy v seznamu (seznam jmen seznamů) se jménem „jména“.

Počáteční uzel, to znamená odstavec, u kterého začíná funkce experta, musí být označen kódem 1. Funkce experta je ukončena odvoláním na kód (odkaz) KONEC, který nemusí být v bázi znalostí definován. Pořadí jednotlivých uzlů (odstavců) v bázi znalostí není rozhodující. Struktura je dána spojením jednotlivých uzlů odkazem na jejich kód.

Bázi znalostí můžeme uložit na disketu do souboru se zadaným jménem. Na disketě je ke jménu souboru automaticky připojena přípona .DAT, kterou při zadávání jména neuvádíme!

Výpis zkušební báze znalostí, stejně jako báze znalostí příkladu „Expertiza při volbě počítače a jeho programového vybavení“ jsou uvedeny ve výpisu.

## Popis funkce programu

Program Expertík umožňuje základní operace pro tvorbu, uložení, výpis a realizaci jednoduché báze znalostí. Základní funkce programu jsou nabídnuty po spuštění programu a v menu obsluhy. Během chodu programu jsou průběžně uváděny nápovědy, které umožňují využít všech jeho funkcí bez podrobného návodu.

Jednotlivé funkce Expertíka nabízené v menu obsluhy jsou následující:

- E** – expertík startuje. Start expertního nebo výukového programu na základě zadané báze znalostí.
- Z** – zápis báze znalostí. Zapsání nové, případně doplnění stávající báze znalostí z klávesnice. Způsob zadání je uveden po vyvolání funkce na obrazovce a je patrný ze zkušební báze a z příkladu báze znalostí.
- V** – výpis báze znalostí. Umožňuje výpis jednotlivých uzlů báze znalostí na obrazovce, případně jejich vymazání z databáze.
- U** – ulož bázi a disketu. Uloží bázi znalostí z operační paměti na disketu do souboru se zadaným jménem.
- N** – nahraj bázi z diskety. Přehraje z diskety do operační paměti bázi znalostí uloženou v souboru se zadaným jménem.
- T** – tiskne bázi na tiskárnu. Vytiskne bázi znalostí z operační paměti na tiskárnu, výpis není stránkovan.
- M** – maz bázi znalostí. Vymaže celou bázi znalostí z operační paměti.
- D** – výpis souboru diskety. Vypíše jména všech souborů diskety.
- K** – konec programu. Ukončí práci programu Expertík s návratem do interpretu jazyka LOGO.

## Popis programu

Program může sloužit jako učebnice jazyka LOGO pro zpracování seznamů. Jednotlivé procedury a funkce je možné po zadání vstupních parametrů volat samostatně a sledovat tak jejich funkci. Proto jsou dále vyjmenovány všechny procedury a funkce, udán popis jejich činnosti a uvedeny případné vstupní parametry.

### Základní procedury

- START** Spuštění programu. Definuje počet znaků jedné řádky obrazovky.  
Volá: POPIS, EXSYS
- POPIS** Vypíše úvodní text s dotazem na nahrání báze znalostí z diskety.  
Volá: ZDISKU
- EXSYS** Realizuje výpis hlavního menu obsluhy a volbu jednotlivých funkcí programu.  
Volá: MENU, VOLBA

- MENU** Vypíše text menu obsluhy na obrazovku.  
Volá: –
- VOLBA** Zajišťuje volbu funkce programu podle zadání obsluhy z klávesnice.  
Volá: ZAPIS, EXPERT, VYPIS, KONEC, NADISK, ZDISKU, MAZE, TISKNE, DISKETA.
- EXPERT** Provádí výpis jednotlivých uzlů báze znalostí podle odpovědi obsluhy na texty spojů (odkazů), realizuje mechanismus vlastního expertního systému.  
Volá: STA, CHYBA
- ZAPIS** Realizuje zápis báze znalostí z klávesnice do operační paměti.  
Volá: ZZ
- VYPIS** Provádí výpis báze znalostí po jednotlivých uzlech na obrazovku, a případně realizuje zadané mazání zobrazeného uzlu.  
Volá: VYPS
- NADISK** Kopíruje bázi znalostí z operační paměti do pojmenovaného souboru na disketě. Ptá se na jméno souboru, po zadání kontroluje jeho existenci. Existuje-li soubor, čeká na odpověď k přemazání stávajícího souboru.  
Volá: WTE
- ZDISKU** Načte obsah zadaného souboru báze znalostí z diskety do operační paměti. Testuje existenci souboru a dává obsluze příslušné nápovědy.  
Volá: RED
- MAZE** Po dotazu provede vymazání celé báze znalostí v operační paměti.  
Volá: MAZ
- TISKNE** Vypíše bázi znalostí z operační paměti na tiskárnu. Dává obsluze příslušné nápovědy.  
Volá: OUT
- DISKETA** Zobrazí na obrazovce seznam souborů na disketě.  
Volá: –
- Pomocné procedury a funkce**
  - STA** Zajišťuje mechanismus výpisu expertního systému až po nalezení kódu uzlu KONEC.  
Vstup: jm kód prvního uzlu  
Volá: VYBER
  - VYBER** Podle kódu najde uzel, který vypíše na obrazovku. Načte

	z klávesnice kód spoje k dalšímu uzlu.		s jméno seznamu Volá: ROT		Vstup: c souřadnice výpisu na obrazovce p počet vypisovaných znaků z vypisovaný znak
<b>PIS</b>	Vypíše číslo a texty spojů (odstavců) nalezeného uzlu.  Vstup: i pořadové číslo spoje v uzlu (objektu seznamu) pr jméno uzlu (seznamu) Volá: ODTAVEC	<b>ROT</b>	Vrací o zadané číslo rotovaný seznam.	<b>SP</b>	Výpis zadaného textu na zvolené místo obrazovky  Vstup: c souřadnice výpisu na obrazovce t jméno seznamu s vypisovaným textem
<b>CHYBA</b>	Výpis echa chyby báze znalostí.	<b>NAJDI</b>	Testuje přítomnost zadaného objektu v seznamu a vrací jeho pořadové číslo v seznamu.  Vstup: p objekt s jméno seznamu Volá: NAJ	<b>ODSTAVEC</b>	Provádí výpis textu ze seznamu do odstavce zadaného souřadnicemi na obrazovce.  Vstup: poz souřadnice na obrazovce seznam jméno seznamu Volá: LINE
<b>ZZ</b>	Realizuje zápis jednoho seznamu (uzlu báze znalostí) z klávesnice nebo z diskety (záleží na výběru vstupního zařízení) až po načtení kódu, ukončujícího zápis.  Vstup: kod kód ukončující zápis Vrací: true byl ukončen zápis Volá: VSTUP	<b>NAJ</b>	Vrací pořadové číslo objektu v zadaném seznamu.  Vstup: c pořadové číslo prvku od kterého probíhá hledání p objekt s jméno seznamu	<b>LINE</b>	Vypíše jeden text seznamu do jednoho řádku odstavce.  Volá: KONRADKU
<b>VSTUP</b>	Zařadí seznam z klávesnice nebo souboru diskety do určeného složeného seznamu.	<b>WTE</b>	Zapiše do souboru na disku seznamy, jejichž jména jsou zadaná v seznamu jmen (uzly báze znalostí).  Vstup: sb jméno souboru na disketě jm jméno seznamu jmen Volá: OUT	<b>KONRADKU</b>	Hlídá fyzický konec řádku na obrazovce.
<b>VYPS</b>	Vypíše jeden uzel na obrazovku a testuje stisk řídicích kláves, podle kterých zruší další výpis, pokračuje výpisem dalšího uzlu, nebo volá proceduru mazající zobrazený uzel (výpis seznamu).  Vstup: jm kód zpracovávaného uzlu (jméno seznamu) Volá: TISK, VYMAZ	<b>OUT</b>	Zapiše na zvolené zařízení seznamy, jejichž jména jsou zadaná v seznamu jmen.  Vstup: j jméno seznamu jmen Volá: OUTS	<b>SAVEC</b>	Pomocná procedura pro zápis zdrojového programu s echem na přepsání existujícího souboru na disk.  Vstup: spec jméno souboru
<b>TISK</b>	Je-li v operační paměti umístěna báze znalostí, vypíše kódem zadaný uzel na obrazovku (vypíše seznam).  Volá: TISKNI	<b>OUTS</b>	Zapiše na zvolené zařízení objekty seznamu (spoje uzlu).  Vstup: s jméno seznamu	<b>Výpis 1. Program Expertik (931-V1)</b>	
<b>TISKNI</b>	Zobrazí na obrazovce jeden uzel (zobrazí objekty seznamu).  Vstup: jm kód uzlu (jméno seznamu) Volá: RADEK	<b>REDA</b>	Čte ze souboru na disketě seznamy, jejichž jména jsou zadaná v seznamu jmen (uzly báze znalostí).  Vstup: sb jméno souboru na disketě Volá: ZAP	<pre> (931-V1) TO START POPIS EXSYS END  TO POPIS TS CT RP [2 8] 19 [*] SP [4 8] [* E X P E R T I K *] RP [6 8] 19 [*] SETCURSOR [8 4] (TYPE "(" "C " ) CHAR 32 [Ing.J.Mitlohner 1988]) SP [10 2] [Program umožňuje tvorbu a spuštění] SP [11 2] [jednoduchého expertního systému.] SP [12 2] [Funkce EXPERTIKA je dana bazi] SP [13 2] [znalosti, kterou muzete vytvorit] SP [14 2] [a zaznamenat na disk. Vytvorenou] SP [15 2] [bazi je mozne využít při novem] SP [16 2] [startu EXPERTIKA.] RP [19 2] 36 [_] SP [21 2] [Chces nahrát ZNALOSTI z diskety ?] SP [22 30] [Ano / Ne] IF RC = "A [ZDISKU] END  TO EXSYS MAKE "k "FALSE MENU VOLBA IF :k [CT SP [2 8] [PROGRAM EXPERTIK JE UKONCEN !] WAIT 40 CT SETCURSOR [1 1] STOP] EXSYS END  TO MENU TS CT </pre>	
<b>RADEK</b>	Vypíše na obrazovku všechny objekty zadaného seznamu.  Vstup: r jméno seznamu	<b>ZAP</b>	Čte ze zvoleného vstupního zařízení seznamy až do nalezení značky EOE (seznamu s textem EOE).  Volá: ZZ		
<b>VYMAZ</b>	Vymaže z operační paměti zadaný seznam (uzel) a zároveň vymaže i jeho jméno ze seznamu kódů uzlů (jmen seznamu).  Volá: DEL	<b>MAZ</b>	Provede vymazání všech proměnných v operační paměti a inicializuje prázdný seznam jmen, vypíše hlášení.		
<b>DEL</b>	Vymaže ze zadaného seznamu označený objekt.  Vstup: p objekt js jméno seznamu Volá: VYPUST, NAJDI	<b>ITM</b>	Vrací objekt se zadaným pořadovým číslem ze zadaného seznamu, při zadání většího čísla než je poslední objekt vrací poslední objekt seznamu.  Vstup: c pořadové číslo objektu v seznamu s jméno seznamu		
<b>VYPUST</b>	Vymaže ze zadaného seznamu objekt se zadaným pořadovým číslem.  Vstup: c pořadové číslo objektu	<b>RP</b>	Provede výpis zvoleného počtu zadaného znaku na zvolené místo obrazovky.		

```

SP [2 5] [MENU - EXPERTIKA]
RP [3 0] 38 [_]
SP [5 5] [E - expertik startuje]
SP [7 5] [Z - zapis baze znalosti]
SP [8 5] [V - vypis baze znalosti]
SP [15 5] [K - konec programu]
SP [9 5] [U - uloz bazi na disketu]
SP [10 5] [N - nahraj bazi z diskety]
SP [11 5] [M - maz bazi znalosti]
SP [12 5] [T - tiskni bazi na tiskarne]
SP [13 5] [D - vypis souboru diskety]
RP [19 0] 38 [_]
SP [21 3] [Stiskni vyznacenu klavesu !]
END

```

```

TO VOLBA
MAKE "z RC
IF :z = "Z [ZAPIS]
IF :z = "E [EXPERT]
IF :z = "V [VYPIS]
IF :z = "K [KONEC]
IF :z = "U [NADISK]
IF :z = "N [ZDISKU]
IF :z = "M [MAZE]
IF :z = "T [TISKNE]
IF :z = "D [DISKETA]
END

```

```

TO EXPERT
CT
IF NOT NAMEP "jmena [MAKE "jmena []]
IF NOT NAMEP "1 [CHYBA] [MAKE "jm 1 STA :jm]
END

```

```

TO ZAPIS
CT PR [ZAPIS BAZI ZNALOSTI VE TVARU :]
RP [1 0] 36 [_] PR []
PR [Kod odstavce]
PR [Nazev odstavce]
PR [[TEXT ODKAZU 1] ODKAZ 1]
PR [[TEXT ODKAZU 2] ODKAZ 2]
PR [a t d dalsi odkazy]
PR [ENTER]
RP [8 0] 36 [_]
RP [22 0] 36 [_]
SP [23 2] [ZAPIS UKONCI DVOJIM STISKEM ENTER]
IF NOT NAMEP "jmena [MAKE "jmena []]
SETCURSOR [9 1]
IF ZZ [] [STOP]
ZAPIS
END

```

```

TO VYPIS
IF NOT NAMEP "jmena [MAKE "jmena []]
VYPS :jmena
END

```

```

TO NADISK
CT SP [8 2] [ZAPIS JMENO BAZE ZNALOSTI]
SETCURSOR [10 2] TYPE [NA DISKETE ! -]
MAKE "soubor RW
IF EMPTY :soubor [STOP]
MAKE "soubor WORD :soubor ".DAT
IF FILEP :soubor [SP [12 2] [BAZE NA DISKETE
EXISTUJE PŘEMAZ Ano / Ne]
IF RC = "A [ERASEFILE :soubor] [STOP]]
SP [14 2] [PROBIHA ZAPIS NA DISKETU !]
WTE :soubor :jmena
END

```

```

TO ZDISKU
CT SP [8 2] [ZAPIS JMENO BAZE ZNALOSTI]
SETCURSOR [10 2] TYPE [NA DISKETE ! -]
MAKE "soubor RW
IF EMPTY :soubor [STOP]
MAKE "soubor WORD :soubor ".dat
IF NOT FILEP :soubor [SETCURSOR [12 2]
(TYPE [BAZE -] :soubor [- NEEEXISTUJE !])
WAIT 30 STOP]
SP [14 2] [PROBIHA CTENI DAT Z DISKETY !]

```

```

RED :soubor
END

TO MAZE
CT SP [8 5] [POZOR ! CHCES OPRAVDU VYMAZAT]
SP [10 5] [VSECHNA DATA ZNALOSTI ? Ano / Ne]
IF RC = "A [MAZ]
END

```

```

TO TISKNE
CT SP [8 2] [ZAPNI A PRIPRAV TISKARNU !]
SP [10 2] [PO TE STISKNI NEKTEROU KLAVESU !]
DRIBBLE "LPT1
MAKE "z RC
SP [14 2] [PROBIHA TISK BAZE ZNALOSTI !]
OUT :jmena
MODRIBBLE
END

```

```

TO DISKETA
CT SP [2 2] [VYPIS SOUBORU Z DISKETY]
SETCURSOR [4 0]
DIR
SP [24 2] [UKONCI STISKEM NEKTERE KLAVESY !]
MAKE :z RC
END

```

```

TO STA :jm
IF EQUALP :jm "KONEC [STOP]
MAKE "jm VYBER :jm
STA :jm
END

```

```

TO VYBER :p
MAKE "pr THING :p
CT PR [] Odstavec CURSOR FIRST :pr
PIS 1 BF :pr
PR [] PR []
REPEAT :sirka [TYPE "-" PR []]
TYPE [Zapis cislo zvoleneho odstavce !]
MAKE "z RC
IF :z = CHAR 27 [OP 1]
IF OR NOT NUMBERP :z :z = 0 [MAKE "z 1]
OP LAST ITM (1 + :z) :pr
END

```

```

TO PIS :i :pr
IF EMPTY BF :pr [STOP]
PR [] PR []
(TYPE "( :i )" "-")
ODSTAVEC CURSOR FIRST FIRST :pr
PIS :i + 1 BF :pr
END

```

```

TO CHYBA
SP [2 10] [ZNALOSTNI BAZE NENI UPLNA !]
WAIT 100
END

```

```

TO ZZ :kod
MAKE "jm RL
IF EQUALP :jm :kod [OP "TRUE]
MAKE "jm FIRST :jm
MAKE :jm []
MAKE "s []
MAKE "jmena FPUT :jm :jmena
VSTUP
REPEAT COUNT :s [MAKE :jm FPUT FIRST :s
THING :jm MAKE "s BF :s]
OP "FALSE
END

```

```

TO VSTUP
MAKE "ss RL
MAKE "s FPUT :ss :s
IF NOT EMPTY :ss [VSTUP]
END

```

```

TO VYPS :jm
CT SETCURSOR [2 1] TISK

```

```

SP [21 2] [Pokracuj stiskem mezerniku !]
SP [22 2] [Ukonci stiskem ESC]
SP [23 2] [Vymaz stiskem Ctrl C]
MAKE "z RC
IF :z = CHAR 3 [VYMAZ]
IF :z = CHAR 27 [STOP]
IF EMPTY :jm [STOP]
VYPS ROT 1 :jm
END

```

```

TO TISK
IF EMPTY :jm [PR [CHYBI DATA ZNALOSTI !]
STOP] [TISKNI :jm]
END

```

```

TO TISKNI :jm
MAKE "prom FIRST :jm
PR :prom
MAKE "rad THING FIRST :jm
RADEK :rad
END

```

```

TO RADEK :r
IF EMPTY :r [STOP]
PR FIRST :r
RADEK BF :r
END

```

```

TO VYMAZ
ERN THING "prom
DEL :prom "jmena
MAKE "jm :jmena
END

```

```

TO DEL :p :js
MAKE "s THING :js
MAKE :js VYPUST NAJDI :p :s :s
END

```

```

TO VYPUST :c :s
MAKE "c :c - 1
OP ROT :c BF ROT (COUNT :s) - :c :s
END

```

```

TO ROT :c :s
REPEAT :c [MAKE "s BL FPUT LAST :s :s]
OP :s
END

```

```

TO NAJDI :p :s
IF NOT MEMBERP :p :s [OP 0]
OP NAJ 1 :p :s
END

```

```

TO NAJ :c :p :s
IF NOT EQUALP :p FIRST :s [MAKE "c NAJ :c
+ 1 :p BF :s]
OP :c
END

```

```

TO WTE :sb :jm
OPEN :sb
SETWRITE :sb
OUT :jm
SHOW [EOB]
SHOW [EOB]
SETWRITE []
CLOSE :sb
END

```

```

TO OUT :j
IF EMPTY :j [STOP]
MAKE "js FIRST :j
PR :js
OUTS THING :js
OUT BF :j
END

```

```

TO OUTS :s
IF EMPTY :s [STOP]

```

```

PR FIRST :s
OUTS BF :s
END

TO RED :sb
MAKE "jmena []
OPEN :sb
SETREAD :sb
ZAP
CLOSE :sb
SETREAD []
END

TO ZAP
IF ZZ [[BOE]] [STOP]
ZAP
END

TO MAZ
ERNS
MAKE "jmena [] MAKE "k "FALSE
MAKE "z "X

CT SP [8 5] [VSECHA DATA ZNALOSTI BYLA
VYMAZANA !]

WAIT 30
END

TO ITM :c :s
MAKE "cou COUNT :s
MAKE "cou :cou - 1
IF :c > :cou [MAKE "c :cou]
IF :c < 1 [MAKE "c 1]
OP ITEM :c :s
END

TO RP :c :p :z
SETCURSOR :c REPEAT :p [TYPE :z]
END

TO SP :c :t
SETCURSOR :c PR :t
END

TO ODSTAVEC :poz :seznam
SETCURSOR :poz

MAKE "cisrad FIRST CURSOR
MAKE "ppoz LAST :poz
REPEAT COUNT :seznam [LINE]
END

TO LINE
IF KONRADKU [SETCURSOR LIST ((FIRST CURSOR)
+ 1) :ppoz]
(TYPE FIRST :seznam CHAR 32)
MAKE "seznam BF :seznam
END

TO KONRADKU
OP (((LAST CURSOR) + (COUNT
FIRST :seznam)) > :sirka)
END

TO SAVEC :spec
IF FILEP WORD :spec ".LF [PR [SOUBOR EXISTUJE,
PREHREJ A / N] IF RC = "A
[ERASEFILE :spec] [STOP]]
SAVE :spec
END

```

## Výpis 2. Báze znalostí (931-V2)

230  
Uz jste nekdy zkousel programovat?  
[Jiste, jsem profesional] 220B  
[Na necem se to prece musim naucit!] 221  
[Nekolik jednoduchych programu jsem vypotil] 221

220B  
Jestli - jste si dostatecne overil sve programatorske  
dovednosti, pak si urcite vyberet sam, podle svych  
financnich moznosti  
[Pocitac kompatibilni s IBM PC XT. Cela rada vyrobcu,  
rozsahle programove vybaveni. Zbyva sehnat software.] 31

221  
Zkuste se nejdrive seznamt s programovanim na levnejsim  
pocitaci v klubech nebo u svych znamych, teprve potom se  
rozhodnete!  
[Pokracuj stiskem nektere klavesy] 17

220AA  
Cela rada domácich 16 bitových pocitacu. Pred koupi nutno  
vlastnosti, cenu, moznosti nakupu konzultovat ( literatura,  
kluby )  
[I pres konzultace si na nakup netroufam] 221  
[Pro zacatek zvolim neco levnejsiho] 210  
[Dekuji, uz jsem si vybral] KONEC

220A  
Ve svete velky vyber vetsinou 16 bitových pocitacu. Pro nase  
podminky nejlepe  
[ATARI ST] 220AA  
[COMMODORE AMIGA] 220AB

220  
Jestli jste jeste nikdy neprogramoval, doporucoji nizsi  
cenovou kategorii. Jinak zalezi na tom, jak chcete pocitac  
vyuzivat.  
[Amatersky pro zabavu a pouceni v domacnosti] 220A  
[Profesionalne pro zhotoveni a prodej programoveho vybaveni]  
220B

210  
K dale nabizenym pocitacum si muzes prikoupit disketovou  
jednotku. Uzitne vlastnosti pocitace se mnohonasobne zvysi!  
[Pokracuj stiskem nektere klavesy] 205

205A  
Ve svete jeden z nejrozsirenejsich a nejlepsich ve sve  
kategorii. U nas dostupny pres inseraty. Malo rozsireny.

[Vyber mi vyhovuje] 202  
[Malo informaci] 203  
[Chci jiny pocitac] 204

205  
Bezny zajem - bezne zbozi  
[COMMODORE 64] 205A  
[Jine pocitace] 201

204  
Chcete obetovat do pocitace vice penez?  
[Ano] 210  
[Ne] 201

203  
Prostudujte si literaturu, nebo se poradte v pocitacovych  
klubech! Vhodne casopisy - ELEKTRONIKA, AMATERSKE RADIO,  
VTM.  
[Pokracujte stiskem nektere klavesy] KONEC

202  
Mnoho uspechu pri koupi pocitace! Chcete poradit pri volbe  
programoveho vybaveni?  
[Ano] 10  
[Ne] KONEC

201D  
U nas pomerne malo rozsireny. Dostupny na inzerat. Omezeno  
programove vybaveni. Dobra grafika a standartni klavesnice.  
[Vyber mi vyhovuje] 202  
[Malo informaci] 203  
[Chci jiny pocitac] 204

201C  
U nas rozsireny pocitac ( TUZEX, maloobchod, inserce )  
[Vyber mi vyhovuje] 202  
[Malo informaci] 203  
[Chci jiny pocitac] 204

201B  
U nas nejrozsirenejsi pocitac. Nevyrabi se. Nestandardni  
klavesnice, bohate programove vybaveni. Dostupny na  
inseraty.  
[Vyber vyhovuje] 202  
[Malo informaci] 203  
[Chci jiny pocitac] 204

201A  
Tuzemsky vyrobek, bezne dosazitelny v obchodech. Temer  
ekvivalent SINCLAIR SPECTRUM. Rozsahle programove vybaveni.  
[Jsem spokojeny s vyberem] 202  
[Malo informaci] 203  
[Chci jiny pocitac] 204

201

Nejcastejsi pripad, ale za malo peněz malo muziky!  
[DIDAKTIK GAMA] 201A  
[SINCLAIR SPECTRUM] 201B  
[ATARI 800 XL / 130 XE] 201C  
[SHARP MZ - 800] 201D

200

Nesnadna odpoved! Kolik chces investovat peněz?  
[Co nejmene] 201  
[Do 10 tis.Kcs nebo 250 DM] 205  
[Do 25 tis.Kcs NEBO 500 DM] 210  
[Do 90 tis.Kcs nebo 1000 DM] 220  
[Nezalezi na penězich] 230

30

Nevzdavejte se tak brzy! I v jiných jazycích musíte umet  
formulovat ulohu, premysleni a drina Vas stejne nemine!  
[Radeji uz toho necham] 19  
[Jeste to zkusim] 52

52

Spravny zacatek Vasi programatorske kariery, ted jde o to  
jak jste si pri tom vedli!  
[Pokracuj stiskem nektre klavesy] 17B

17B

S jakymi vysledky jsi uspel v programovani?  
[Dokazi zapsat jednoduche programy, mam zajem o programovani  
v jinem jazyce] 30  
[Samostatne napisi a odladim i slozite programy vyuzivajici  
zapisu a vypisu dat na disketu ( kazetu )] 31  
[Pres vsechnu snahu mi dela potize sestaveni i jednoduchych  
programu] 19

17A

LOGO je rozsirene na vetsine typech pocitacu. Obrat se na  
pocitacovy klub, stanice mladych techniku a pod. zarizeni!  
[Pokracuj stiskem nektre klavesy!] 17

17

Pro vstup do sveta programovani doporučuji jazyk LOGO. Je  
moderni a vytvorite si dobry zaklad pro vyuku dalsich  
jazyku.  
[Nemohu LOGO na svůj pocitac sehnat] 17A  
[Seznamil jsem se s jazykem LOGO] 17B

51

Nedostal jste se sice daleko, ale jste na spravne ceste  
[Pokracuj stiskem nektre klavesy!] 17

50

Nezvolil jste dobry vstup do zivota programatora! Radeji  
skuste rychle jeste neco jineho!  
[Pokracuj stiskem nektre klavesy] 17

20

Chcete mou radu, pak nejste s programovanim prilis daleko.  
V jakem jazyku jste prevazne programoval?  
[BASIC] 50  
[KAREL] 51  
[LOGO] 52  
[Jednim, nebo vice z PASCAL, PROLOG, C, LISP, ASSEMBLER, COBOL]  
31  
[V zadnem a nebo ve vseh, ale ne moc uspesne] 17

44

Krome specializovanych prostredku jazyky PROLOG, LISP,  
MAMPS, C  
[Ukonci stiskem nektre klavesy] KONEC

43

Cela rada specialnich programovych prostredku na pr.  
AUTOCAD, CHART, DRHALLO, ORCAD a jine. Doplnit jazyky C, LISP,  
pr. PASCAL.  
[Ukonci stiskem nektre klavesy!] KONEC

42

Integrovaný software na pr. dBASE, FRAMEWORK, SUPERCALC  
a jine. Z jazyku pak C - jazyk, PASCAL  
[Ukonci stiskem nektre klavesy!] KONEC

41

Existuje bohata nabidka speciálního vybaveni na  
pr. STATGRAPHICS, EUREKA a dalsi. Jinak jazyky FORTRAN,  
PASCAL, BASIC  
[Ukonci stiskem nektre klavesy!] KONEC

40

Drive bych doporučil ASSEMBLER. Ten je vsak zavisly na typu  
pocitace. Nyni jednoznacne doporučuji C - jazyk !  
[Ukonci stiskem nektre klavesy!] KONEC

31

Mate talent pro programovani! Dale zalezi na oboru, ve  
kterem tento talent chcete uplatnit  
[Rizeni stroju, robotu, technologii] 40  
[Vedecko technicke vypocty, matematika] 41  
[Ekonomicke vypocty, administrativa] 42  
[Projektovani, pocitacova grafika] 43  
[Tvorba zakladního programového vybaveni] 40  
[Expertni systemy, znalostni systemy, umela inteligence] 44

19

Programator z Vas nebude, ted uz nemuzete nic pokazit,  
zkuste jazyk BASIC, mozna ze Vas uspokoji!  
[Ukonci stiskem nektre klavesy] KONEC

21

Pokuste se sehnat programovací jazyk LOGO, ale nenechte se  
na zacatku odradit neuspechem! Vytrvejte!  
[Ukonci stiskem nektre klavesy] KONEC

18

Zkuste nejdrive programovací jazyk KAREL  
[Programovani se Vam darilo a bavi Vas?] 17  
[Hra s robotem me nezaujala, ale programovat bych chtel  
v jinem jazyce] 21  
[Robot me nechce poslouchat, jeho slozitejsi cinnost nedokazi  
formulovat] 19  
[Nesehnal jsem program KAREL na muj pocitac] 21

1

Mate problem s vyberem pocitace a programoveho vybaveni?  
Expertik Vam pomuze! Staci odpovidat na jeho dotazy a poradit  
Vam!  
[Chci vyuzit sluzeb expertika pro vyber programoveho  
vybaveni] 10  
[Chci si poridit svůj první pocitac] 200  
[Nemam zajem o pomoc expertika] KONEC

10

Vyber zavisí na Vasich zajmech, dosavadnich znalostech  
a typu pocitace.  
[Nevim jeste, jaky pocitac si poridim] 200  
[Pocitac jsem si uz vybral, nemam zadne zkušenosti  
s obsluhou a programovanim] 15  
[Mam pocitac a znam zaklady programovani, chci se  
programovani systematicky venovat] 20

15

Musite se rozhodnout k jakym ucelum chcete Vas pocitac  
prevazne vyuzivat.  
[Hry a nebo hotove, jednocelove programy] 16  
[Mam zajem se naucit programovat] 17  
[Jeste nevim, chci se rozhodnout pozdeji] 18

16

Hotove programy a hry Vam nabidnou pocitacove kluby, stanice  
mladych techniku, domy mladeze a podobne.  
[O jine programy uz nemam zajem, nebo az pozdeji] KONEC  
[Prece jenom bych si rad zkusal neco sam naprogramovat] 18

[EOE]

[EOE]

# INTERFEJS TISKÁRNA – ATARI

Ing. J. Kodera, Hůrka 1058, 278 01 Kralupy n. V.

**Možnost připojení tiskárny je jednou ze základních podmínek pro skutečně efektivní využití mikropočítačů všech typů. U nás stále rozšířenější osmibitové domácí mikropočítače ATARI umožňují připojení různých periferních zařízení (disketová jednotka, tiskárna, datový magnetofon), přes sériový port. Tento port však bohužel neodpovídá žádnému běžnému standardu a tak lze bez problémů připojit pouze periférie dodávané přímo firmou nebo vybavené příslušným interfejsem (z tiskáren např. ATARI 1027 nebo 1029, SEIKOSHA GP500AT). (Zapojení bylo vyvinuto a zasláno redakci v roce 1987 – pozn. red.)**

Běžnou tiskárnu vybavenou např. rozhraním Centronics lze v zásadě připojit dvěma způsoby:

1. Využití paralelní port počítače používaný jinak pro připojení ovladačů (joysticků), obsluhovaný v počítači obvodem PIA 6520, což znamená napsat příslušný obslužný program pro počítač. Nevýhodou tohoto řešení je, že není podporováno žádným z běžného programového vybavení.

2. Zkonstruovat interfejs, který se připojí mezi sériový port a tiskárnu. Toto řešení je sice náročnější, zato však daleko pružnější a zachovává všechny výhody počítače včetně podpory operačního systému, BASICu a veškerého softwaru (např. textových editorů).

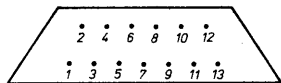
Druhé řešení je obsahem následujícího příspěvku.

## Princip komunikace počítače s periferními zařízeními

Veškeré periférie se k počítači připojují přes třináctipólový konektor, umístěný na zadní straně počítače. Popis konektoru je na obr. 1. Pro připojení interfejsu se využívají pouze vývody č. 3, 4, 5, 7, 10. Periférie (s výjimkou magnetofonu) jsou vybaveny vlastní „inteligencí“ a jsou připojeny přes budiče s otevřenými kolektory, takže může být paralelně připojeno více zařízení najednou, aniž by se vzájemně ovlivňovala. Přesto jsou standardní periférie vybaveny dvěma konektory podle obr. 1 propojenými paralelně.

Všechny napěťové úrovně jsou TTL, logika je pozitivní.

Komunikace je sériová asynchronní, značky jsou osmibitové s jedním START bitem (log.0) a jedním STOP bitem (log.1).



Obr. 1. Konektor počítačů Atari pro sériový styk s perifériemi (901–1). Popis vývodů:

- |                 |  |
|-----------------|--|
| 1 CLOCK IN      | nepoužívá se                                   |
| 2 CLOCK OUT     | nepoužívá se                                   |
| 3 DATA IN       | nepoužívá se                                   |
| 4 GND           | zem  |
| 5 DATA OUT      | seriová vstupní datová linka                   |
| 6 GND           | zem  |
| 7 CMDn          | seriová výstupní datová linka                  |
| 8 MOTOR CONTROL | označení povelu řízení motoru dat. magnetofonu |
| 9 PROCEED       | nepoužívá se                                   |
| 10 + 5 V        | napájecí napětí                                |
| 11 AUDIO IN     | vstup akustického signálu                      |
| 12 NC           | nepoužívá se                                   |
| 13 INTERRUPT    | nepoužívá se                                   |

Rychlost je 19 200 Bd, používá se standardní soubor znaků ASCII, přičemž znaky zobrazované na stínítku inverzně mají nastavený nejvyšší bit do log.1. V klidu je na kontaktech (samozřejmě s výjimkou GND) úroveň log.1.

Jak již bylo řečeno, komunikace je „inteligentní“. V praxi to vypadá např. po příkazu LIST „P:“ následovně:

1. Počítač oznámí všem připojeným periferním zařízením logickou úroveň 0 na vývodu CMD, že vydává blok povelu. Zároveň vyšle pět povelových bajtů na vývod DATA OUT. Složení povelového bloku je: jeden bajt označení periférie (40H pro tiskárnu, „P:“), jeden bajt označení povelu (53H je žádost o status), dva bajty dodatečné informace (pro tiskárnu dvakrát 0), jeden bajt zabezpečení (součet všech čtyř předešlých bajtů s uvážením přenosu např. instrukci ADC většiny mikroprocesorů).
2. Po skončení signálu CMD odpoví oslovená periférie přes vývod DATA IN podle druhu povelu. Odpověď na žádost o status je sedmibajtová.
3. Počítač vydá další povelový blok, opět zároveň se signálem CMD. Tento blok je stejný jako v bodě 1, pouze je jiné označení povelu (57H pro zápis dat na periferní zařízení s kontrolním součtem).
4. Oslovená periférie opět odpoví, tentokrát jen jedním bajtem (41H jako kladné potvrzení povelu).
5. Počítač vydá jeden blok dat, standardně dlouhý 40 datových bajtů plus jeden bajt zabezpečení (stejně jako u povelového bloku).

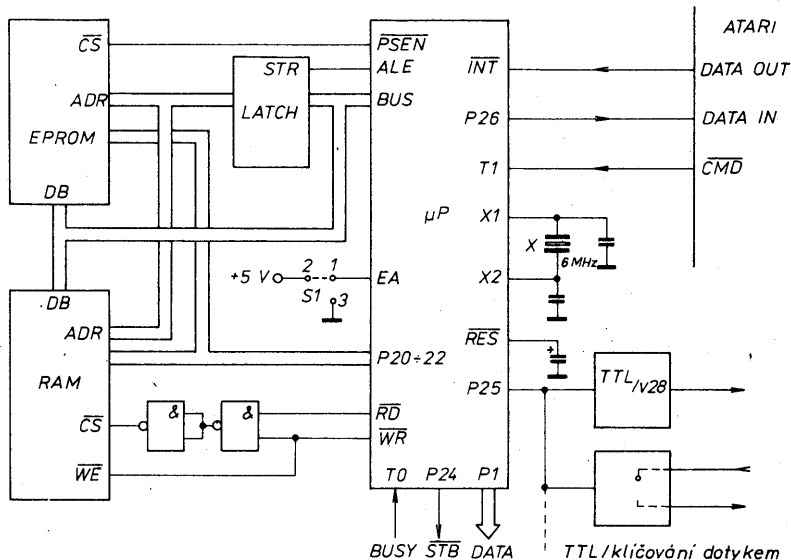
6. Periférie odpoví jedním bajtem (41H jako kladné potvrzení), zpracuje přijatá data a oznámí počítači, že komunikace může pokračovat (bajtem 43H).

7. Není-li komunikace ukončena, pokračuje počítač bodem 3.

V průběhu komunikace může samozřejmě dojít k poruše; pokud počítač nedostane včas odpověď na svůj povel, opakuje ho (standardně celkem 13×), pokud ani jednou přitom nedostane odpověď, vyhlásí chybu č. 138 (TIME OUT). Stejně je tomu v případě, že počítač nedostane kladné potvrzení o přijetí dat.

## Realizace a stavba zařízení

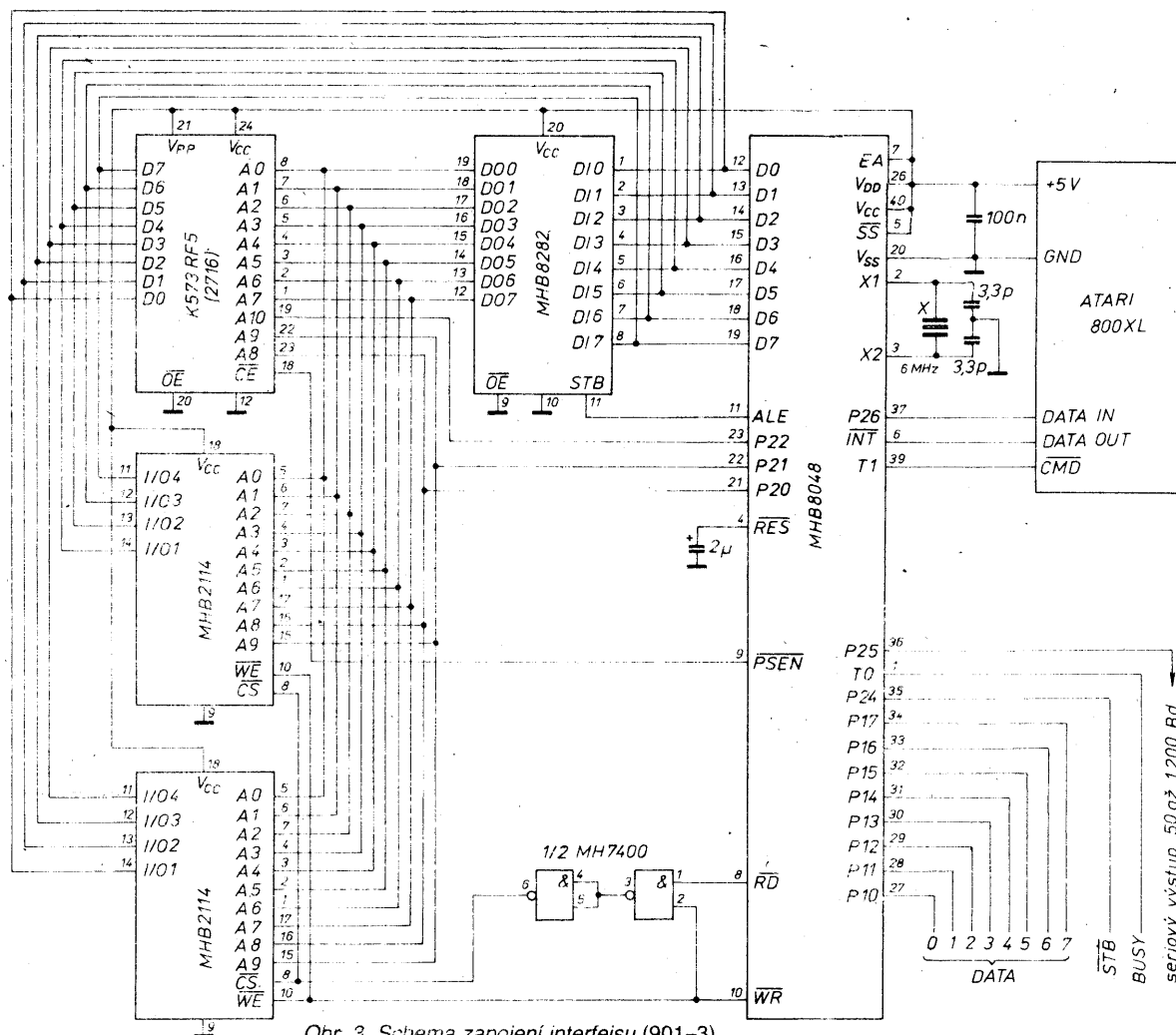
Z uvedeného popisu způsobu komunikace počítače s perifériemi vyplývá, že realizace interfejsu je možná prakticky jen při použití mikroprocesoru, který je řízen programem zajišťujícím všechny potřebné funkce a který zároveň umožňuje interfejs přizpůsobit prakticky libovolné tiskárně. Jakékoli čistě hardwarové řešení by vyšlo neúměrně složité a nákladné. Jako prakticky ideální řešení se nabízí použití některého jednočipového mikropočítače z řady 48, který ke své funkci potřebuje minimum dalších součástek a je relativně dostupný i v ČSSR. Principiální schéma interfejsu je na obr. 2. Pokud použijeme mikropočítač 8048 nebo 8035, je třeba připojit obě vnější paměti (RAM i EPROM). Výhodnější je použít mikropočítač 8747 se zabudovanou vnitřní pamětí EPROM, po-



Obr. 2. Blokové schéma zapojení interfejsu (901–2).

Propojka S1 je pro typ 8748 v poloze 1–3, pro ostatní typy v poloze 1–2.





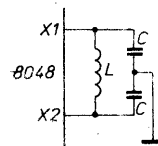
Obr. 3. Schema zapojení interfejsu (901-3)

tom odpadá vnější programovatelná paměť, avšak tento obvod je hůře dostupný a pro většinu amatérů je velmi obtížné zajistit naprogramování vnitřní paměti. Výhodné je i použití mikropočítače 8039, který má postačující kapacitu vnitřní RAM, s dostupností tohoto obvodu je to však ještě horší (k. p. Tesla zatím dodává typy 8048, 8035 a 8748).

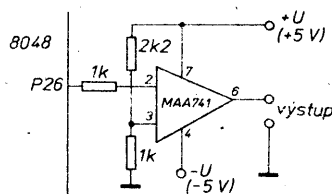
Na místě vnější RAM je možné použít libovolnou paměť RAM s kapacitou alespoň 64 bajtů, vyhoví např. dva tuzemské obvody MHB2114 s organizací  $1k \times 4$  byty, ze zahraničních typů např. 6116 (s kapacitou 2 kB) apod. Latch je osmibitový paralelní strádač, např. MHB8282, SN74373, apod. EPROM je libovolná s kapacitou alespoň 1 kB – např. MHB8708C, výhodnější z hlediska zachování jediného napájecího napětí je však typ 2716, resp. sovětský ekvivalent K573RF5.

Jedna z možností konkrétní realizace je na obr. 3. Jsou zde použity obvody, o kterých se domnívám, že by měly být nejsnáze dostupné. Jak však bylo řečeno, možnosti je mnoho a záleží na možnostech a na štěstí každého zájemce o stavbu interfejsu, které konkrétní obvody se mu podaří sehnat. Z tohoto důvodu rovněž neuvádím výkres desky s plošnými spoji; navíc se domnívám, že pro stavbu jednoho kusu je nejvýhodnější využít některou univerzální zapojovací desku a drátové propojení. Výhodné je však použít sokly pro mikroprocesor a samozřejmě pro EPROM.

Kritickým místem celé konstrukce je zřejmě krystal. Díky velmi vysoké přenosové rychlosti bylo nutno časování odvodit přímo z doby trvání instrukčního cyklu, která je pro krystal 6 MHz 2,5  $\mu$ s. Odpovídající kmitočet je pro základní varianty mikropočítačů řady 48 nejvyšší dovolený, pro vyšší kmitočty by



Obr. 4. Náhrada krystalu 6 MHz (901-3)



Obr. 5. Jednoduchý převodník TTL/V.28 (901-5)

navíc bylo třeba modifikovat řídicí program. Při nižším kmitočtu by již mikroprocesor nestihl obsluhovat příjem dat z počítače.

Výrobce dovoluje namísto krystalu zapojit i obvod podle obr. 4. Při přesném vyladění kmitočtu na 6 MHz by neměla být funkce interfejsu narušena. Problémy by však mohly nastat s časovou a teplotní stabilitou. Toto řešení nebylo experimentálně ověřeno.

Pro ty, kdo chtějí interfejs využít jako převodník na sériové rozhraní s napětovou úrovní podle doporučení CCITT V.28 (RS 232C), je nutné ještě zapojení vybavit převodníkem TTL/V.28, např. podle obr. 5. Jiná řešení převodníku lze najít v [3]. Přitom je

nutno zapojení ještě doplnit o záporné napájecí napětí (lze je získat např. z tiskárny). Optimální by samozřejmě bylo napájení +12 V, protože však doporučení V.28 definuje jako zakázanou oblast -3V až +3V, lze bez problémů použít napájení 5V.

Interfejs lze napájet přímo z počítače napětím +5V (výrobce udává proudovou „rezervu“ zdroje větší než 0,5 A), díky oddělenému napájecímu zdroji nehrozí poškození počítače ani při náhodném zkratu.

Protože originální třináctipólový konektor je téměř nesehnatelný (nejen u nás), je nutné tento konektor nahradit. Jako kontakty vyhoví např. dutinky z rozebraného konektoru CANON DB 25 (standard pro rozhraní V. 24) vlepené do destičky z organického skla s předvrtanými otvory v rastru 3,5 mm. Ještě lepší je však zabudovat do počítače např. pětikolíkový nf konektor (DIN), paralelně uvnitř počítače propojený na příslušné špičky třináctipólového konektoru. Tento konektor pak lze používat i pro připojení upraveného magnetofonu atd. Na straně tiskárny použijeme konektor podle typu tiskárny.

## Řídicí program

Obsah EPROM s řídicím programem pro jednočipový mikropočítač je ve **Výpisu 1** (od adres nutno odečíst offset 8000H), obsah upraveného programu pro mikroprocesor 8039 (zapojení bez vnější RAM), je ve **Výpisu 2**.

Každý bajt z počítače je přijat rutinou pro zpracování interruptu – je na adresách 0AH až 40H. Po přijetí povelového bloku mikroprocesor zjistí, zda je oslovena tiskárna a pokud ano, vyšle na počítače příslušnou odpo-

věd přes vývod P26 a vyčká na datový blok. Ten je ukládán do RAM, poté se zjistí případný výskyt řídicích posloupností pro interfejs (viz dále) a nakonec je blok přenesen bajt po bajtu na port P1, který tvoří rozhraní CENTRONICS (porty 8048 jsou zakončeny budiči s otevřeným kolektorem a střádačem, což je pro tento účel výhodné). Vývod P24 mikroprocesoru se využívá pro výskyt potvrzovacích impulsů STB, vývod TO pro příjem signálu BUSY z rozhraní. Tyto dva signály spolu s osmi datovými budiči tvoří základní a postačující obvody rozhraní CENTRONICS. Je-li naprogramováno sériové vysílání, jsou bajty sériově vysílány přes vývod P25 zadanou přenosovou rychlostí. Po vyslání celého bloku interfejs oznámí počítači možnost pokračovat vysláním bajtu 43H na vývod P26. Je-li uvnitř přijatého bloku nalezen znak EOL, tj. 9BH, je nahrazen kombinací CR+LF (0DH+0AH). EOL je totiž označení konce logického řádku používané počítači ATARI. Protože počítač doplní blok obsahující

její EOL samými nulami až do 40 bajtů, jsou tyto nevýznamné nuly interfejsem ignorovány.

Máme-li k dispozici maticovou tiskárnu, která má sedm až devět jehliček a která využívá posloupnost ESC-K k přecházení do grafického režimu (např. tiskárny EPSON a kompatibilní, novější maticové tiskárny ROBOTRON atd.), můžeme využít další funkce interfejsu. Při příjmu znaků s ASCII kódem 0 až 19H (0 až 31 dekadicky) umožňuje interfejs buď transparentní přenos těchto znaků (např. ASCII znak 14H=CTRL-N se používá pro tisk jednoho řádku zvětšenými písmeny), nebo přepne tiskárnu do grafického režimu a vytiskne přibližně stejný znak, jaký je zobrazen na stínítku (ATARI pseudografika). Každý znak je reprezentován šesti bajty – tzn. že je tisknou v matici 6×8, stejně jako alfanumerické znaky u tiskáren EPSON, na rozdíl od matice 8×8 na stínítku. Kódy pro grafické znaky jsou uloženy ve třetí stránce EPROM tak, že na adresách 300H

až 305H je šest bajtů pro znak s kódem 1H atd. Tuto tabulku lze samozřejmě při programování EPROM předdefinovat a vytvořit tak vlastní znaky, např. českou abecedu s háčky a čárkami. Spolu s možností vytvoření uživatelské znakové sady, kterou poskytuje počítač ATARI, máme tak účinný prostředek např. pro vytvoření vlastního textového editoru s českou abecedou a s možností tisku.

Přitom v grafickém režimu je možno přepsat, že se znak ESCAPE (kód 1BH) tiskne jako grafický znak nebo se interpretuje jako řídicí znak, což umožňuje používat řídicí posloupnosti pro tiskárnu (např. k přepínání druhů písma). V transparentním režimu je naopak možno přepsat, že interfejs všechny znaky s ASCII kódem 0 až 19H nahradí znakem ~ (kód 7EH), pokud je transparentní přenos nežádoucí.

Pro řízení interfejsu se využívá některých posloupností se znakem ESCAPE, které ne-

```
8000 44 BA 20 15 04 0A 31 37 38 20 C5 AE 00 00 00 27
8010 AB 86 15 04 3B BC 08 B9 03 46 21 BF 05 B8 20 04
8020 27 D5 BD 29 B8 00 C5 E9 27 97 86 2F A7 04 32 00
8030 00 00 FB 67 AB B9 03 00 EC 27 A5 FE 86 3C 05 93
8040 80 D3 72 96 51 18 FF 53 F8 AF 80 53 F8 5F AF 04
8050 9E 80 D3 6A 96 5C FF 53 F7 AF 04 9E 80 D3 69 96
8060 67 FF 43 08 AF 04 9E 80 D3 63 96 72 FF 43 40 AF
8070 04 9E 80 D3 7A 96 7D FF 53 EF AF 04 9E 80 D3 64
8080 96 88 FF 43 20 AF 04 9E 80 D3 6E 96 93 FF 53 DF
8090 AF 04 9E 80 D3 67 C8 96 A1 18 FF 53 BF AF 97 A7
80A0 83 97 83 2E 42 59 54 55 0D 0A 20 20 30 31 33 38
80B0 20 42 45 34 31 20 20 20 20 20 20 20 20 31 38
80C0 32 20 20 20 20 20 20 20 20 20 20 20 20 32 36
80D0 2C 23 34 31 48 0D 0C 0A 0A 49 53 49 53 2D 49
80E0 49 20 4D 43 53 2D 34 38 2F 55 50 49 2D 34 31 20
80F0 4D 41 43 52 4F 20 41 53 53 45 4D 42 4C 45 52 2C
8100 76 0F B5 C5 56 30 FB A0 18 EF 00 34 9E 96 00 46
8110 0F 85 BF 05 23 20 A8 D5 17 AF F1 D3 53 C6 25 BE
8120 41 54 95 24 00 B8 F8 B9 07 54 8C B8 00 95 24 00
8130 FB D5 90 18 ED 00 BD 29 BE 41 B6 00 95 BB FF EB
8140 3F 54 95 B8 00 B9 28 BA 10 9A F8 80 D3 9B C6 94
8150 FF 37 92 62 80 D3 1B 96 62 18 14 40 E6 62 E9 8F
8160 24 96 FF D2 7F B2 6C 80 D3 1B C6 8C 54 4B F6 8F
8170 80 53 7F 03 E0 F6 8C 54 61 54 71 54 7F 24 8F FF
8180 B2 8C 80 53 7F 03 E0 F6 8C 23 7E 90 80 54 00 18
8190 E9 4B 24 96 54 42 BE 43 54 95 B8 00 24 00 BF 05
81A0 B8 20 F0 D3 40 83 20 53 55 50 45 52 20 20 20 20
81B0 20 20 20 3B 46 30 3D 4E 45 4E 49 20 54 4F 20 50
81C0 52 4F 20 4C 50 3A 0D 0A 20 20 30 31 33 43 20 39
81D0 35 20 20 20 20 20 20 20 20 20 20 20 31 38 34 20
81E0 20 20 20 20 20 20 20 20 20 43 50 4C 20 46 30 20 20
81F0 20 20 20 20 20 20 20 20 20 3B 4A 45 20 54 4F 20 50
8200 AE FF 72 08 80 53 7F 90 FF 53 07 96 16 36 0D FE
8210 39 9A EF 8A 10 83 FE 35 9A DF 54 37 BA 08 00 00
8220 67 F6 27 9A DF 44 2B BA 20 00 00 54 37 EA 20 00
8230 00 00 8A 20 54 37 83 AE FC 62 55 FE 16 40 44 3C
8240 65 83 23 0D 54 00 23 0A 54 00 83 97 80 D3 A0 96
8250 60 FF 37 72 60 A7 54 61 23 FF BB 06 54 00 EB 5C
8260 83 23 1B 54 00 23 4B 54 00 23 06 54 00 27 54 00
8270 83 80 97 85 F7 F6 78 95 AB E7 6B BB 06 AD 83 FD
8280 E3 B6 84 37 54 00 1D EB 7F BD 29 83 F8 A3 AE 54
8290 95 18 E9 8C 83 9A BF BB 04 00 EB 9A BA 08 FE BB
82A0 05 67 F6 A8 9A BF 44 AC 8A 40 00 00 EB AC EA 9F
82B0 00 00 00 BB 08 8A 40 EB B7 63 15 23 70 3A 27 C5
82C0 BF 05 AE AD AC AB AA A9 B8 20 D5 AF AE BD 29 AC
82D0 AB AA A9 A3 8A 80 09 AF 9A 7F 85 95 A5 B5 35 FF
82E0 53 07 03 F0 A3 AC C5 05 24 00 2E 41 44 52 45 53
82F0 00 07 84 C2 D7 EC F6 00 41 43 00 00 14 29 3D 20
8300 00 38 7C 3E 7C 38 00 00 FF FF 18 18 00 00 00 00
8310 FF FF 18 18 F8 F8 00 00 18 18 FF FF 00 00 18 18
8320 1F 1F 00 00 07 0E 1C 38 70 E0 E0 70 38 1C 0E 07
8330 03 07 0F 1F 3F FF 00 00 0F 0F 0F FF 3F 1F 0F
8340 07 03 00 00 00 F0 F0 F0 F0 F0 F0 F0 00 00 C0 C0
8350 C0 C0 C0 C0 03 03 03 03 03 F0 F0 F0 00 00 00 00
8360 00 1D 77 63 77 1D 00 00 1F 18 18 18 18 18 18
8370 18 18 18 18 FF FF 18 18 00 18 3C 3C 18 00 0F 0F
8380 0F 0F 0F 0F FF FF FF 00 00 18 18 1F 1F 18 18
8390 18 18 F8 F8 18 18 FF FF FF 00 00 00 00 00 F8 F8
83A0 18 18 00 7C 54 47 05 00 00 18 30 7E 30 18 00 18
83B0 0C 7E 0C 18 08 08 2A 3E 1C 08 08 1C 3E 2A 08 08
83C0 56 20 52 41 4D 0D 0A 20 20 30 31 34 42 20 38 30
83D0 20 20 20 20 20 20 20 20 20 20 20 31 39 32 20 54
83E0 49 53 4B 31 3A 20 20 2D 4F 56 58 20 41 2C 40 52
83F0 30 0D 0A 20 20 30 31 34 43 20 44 33 39 42 20 20
```

```
8000 44 BA 8D 15 04 0A 36 92 1F D2 C5 AE 00 00 00 27
8010 AB 86 15 04 3B BC 08 B9 03 46 21 BF 05 B8 20 04
8020 27 D5 BD 29 B8 40 C5 E9 27 97 86 2F A7 04 32 00
8030 00 00 FB 67 AB B9 03 00 EC 27 A5 FE 86 3C 05 93
8040 F0 D3 72 96 51 18 FF 53 F8 AF F0 53 F8 5F AF 04
8050 9E F0 D3 6A 96 5C FF 53 F7 AF 04 9E F0 D3 69 96
8060 67 FF 43 08 AF 04 9E F0 D3 63 96 72 FF 43 40 AF
8070 04 9E F0 D3 7A 96 7D FF 53 EF AF 04 9E F0 D3 64
8080 96 88 FF 43 20 AF 04 9E F0 D3 6E 96 93 FF 53 DF
8090 AF 04 9E F0 D3 67 C8 96 A1 18 FF 53 BF AF 97 A7
80A0 83 97 83 7E 22 BA A8 CD 5F 7F C9 21 AE 8D 22 B7
80B0 A8 2B 7E 32 B9 A8 11 C3 8D 01 68 7E CD BD 84 11
80C0 C3 8D 01 AD 8D CD E8 83 FE 00 C2 09 31 21 68 A8
80D0 36 03 11 39 92 01 C4 8D CD 32 6C 3E 00 11 39 92
80E0 CD 08 8C B5 D6 01 9F F5 3E 0D 1B CD 08 8C B5 D6
80F0 01 9F C1 48 B1 1F D2 06 81 01 AE 8D C5 11 39 92
8100 76 0F B5 C5 56 30 FB A0 18 EF 00 34 9E 96 00 46
8110 0F 85 BF 05 23 20 A8 D5 17 A9 F1 D3 53 C6 25 BE
8120 41 54 95 24 00 B8 F8 B9 07 54 8C B8 00 95 24 00
8130 FB D5 A0 18 ED 00 BD 29 BE 41 B6 00 95 BB FF EB
8140 3F 54 95 B8 40 B9 28 BA 10 9A F8 F0 D3 9B C6 94
8150 FF 37 92 62 F0 D3 1B 96 62 18 14 40 E6 62 E9 8F
8160 24 96 FF D2 7F B2 6C F0 D3 1B C6 8C 54 4B F6 8F
8170 F0 53 7F 03 E0 F6 8C 54 61 54 71 54 7F 24 8F FF
8180 B2 8C F0 53 7F 03 E0 F6 8C 23 7E A0 F0 54 00 18
8190 E9 4B 24 96 54 42 BE 43 54 95 B8 40 24 00 BF 05
81A0 B8 20 F0 D3 40 83 23 72 0E 06 CD 14 7E 3E 00 11
81B0 39 92 CD 08 8C B5 CA BA 81 C9 2A C4 A8 E5 2A A7
81C0 8D 44 4D 2A 9D 8D EB CD 2A 7E 3E 00 11 39 92 CD
81D0 08 8C B5 CA D7 81 C9 2A C5 A8 2B 22 C5 A8 C3 88
81E0 81 CD 81 72 C9 21 C7 A8 71 2A C7 A8 4D 1E 02 CD
81F0 3C 7D 3E 00 11 39 92 CD 08 8C B5 CA FF 81 C9 2A
8200 AE FF 72 08 F0 53 7F A0 FF 53 07 96 16 36 0D FE
8210 39 9A EF 8A 10 83 FE 35 9A DF 54 37 BA 08 00 00
8220 67 F6 27 9A DF 44 2B BA 20 00 00 54 37 EA 20 00
8230 00 00 8A 20 54 37 83 AE FC 62 55 FE 16 40 44 3C
8240 65 83 23 0D 54 00 23 0A 54 00 83 97 80 D3 A0 96
8250 60 FF 37 72 60 A7 54 61 23 FF BB 06 54 00 EB 5C
8260 83 23 1B 54 00 23 4B 54 00 23 06 54 00 27 54 00
8270 83 80 97 85 F7 F6 78 95 AB E7 6B BB 06 AD 83 FD
8280 E3 B6 84 37 54 00 1D EB 7F BD 29 83 F8 A3 AE 54
8290 95 18 E9 8C 83 9A BF BB 04 00 EB 9A BA 08 FE BB
82A0 05 67 F6 A8 9A BF 44 AC 8A 40 00 00 EB AC EA 9F
82B0 00 00 00 BB 08 8A 40 EB B7 63 15 23 70 3A 27 C5
82C0 BF 05 AE AD AC AB AA A9 B8 20 D5 AF AE BD 29 AC
82D0 AB AA A9 A3 8A 80 09 AF 9A 7F 85 95 A5 B5 35 FF
82E0 FF 53 07 03 F0 A3 AC C5 05 24 00 2E 41 44 52 45 53
82F0 00 07 84 C2 D7 EC F6 00 41 43 00 00 14 29 3D 20
8300 00 38 7C 3E 7C 38 00 00 FF FF 18 18 00 00 00 00
8310 FF FF 18 18 F8 F8 00 00 18 18 FF FF 00 00 18 18
8320 1F 1F 00 00 07 0E 1C 38 70 E0 E0 70 38 1C 0E 07
8330 03 07 0F 1F 3F FF 00 00 0F 0F 0F FF 3F 1F 0F
8340 07 03 00 00 00 F0 F0 F0 F0 F0 F0 F0 00 00 C0 C0
8350 C0 C0 C0 C0 03 03 03 03 03 F0 F0 F0 00 00 00 00
8360 00 1D 77 63 77 1D 00 00 1F 18 18 18 18 18 18
8370 18 18 18 18 FF FF 18 18 00 18 3C 3C 18 00 0F 0F
8380 0F 0F 0F 0F FF FF FF 00 00 18 18 1F 1F 18 18
8390 18 18 F8 F8 18 18 FF FF FF 00 00 00 00 00 F8 F8
83A0 18 18 00 7C 54 47 05 00 00 18 30 7E 30 18 00 18
83B0 0C 7E 0C 18 08 08 2A 3E 1C 08 08 1C 3E 2A 08 08
83C0 39 92 CD 08 8C B5 C2 CC 83 CD A7 82 CD 2B 60 C3
83D0 DB 83 CD 95 84 CD A7 82 CD 2B 60 C9 0A 3C C5 42
83E0 4B 5F 16 00 CD FE 8A C9 EB 0A 5F 1C 0B 2C 23
83F0 0A BE C2 FC 83 1D C2 EE 83 3E FF C9 AF C9 2A B5
```

jsou běžnými tiskárnami využívány. Jejich přehled spolu s významem je v tabulce 1.

Po zapnutí je interfejs inicializován tak, že je transparentní pro všechny znaky a výstup je na port T1, tj. rozhraní CENTRONICS. Chceme-li, aby výstup z interfejsu byl sériový rychlostí např. 50 Bd, aby znaky s kódem menším než 20H byly tisknuty v grafickém režimu včetně znaku ESCAPE, a inverzní alfanumerické znaky mají být tisknuty jako normální (nejvyšší bit nastaven do 0), zadáme jako první příkaz po zapnutí počítače:

```
LPRINT „ESC·r“; CHRS (1);
„ESC·j·ESC·d·ESC·g“
```

Chceme-li, aby interfejs pracoval v režimu CENTRONICS, přičemž znaky s kódem nižším než 20H mají být nahrazeny vlnovkou a interfejs má v tomto režimu trvale zůstat, zadáme po zapnutí:

```
LPRINT „ESC·r“; CHRS (0);
„ESC·n·ESC·c·ESC·z“
```

Konečně chceme-li, aby interfejs pracoval v režimu CENTRONICS, přičemž znaky s kódem nižším než 20H mají být interpretovány jako grafické a zároveň chceme mít možnost používat řídicí posloupnosti pro tiskárnu, tzn. že ESCAPE nesmí být interpretován jako grafický znak, zadáme:

```
LPRINT „ESC·r“; CHRS (0);
„ESC·n·ESC·g“
```

Podobných příkladů lze samozřejmě vymyslet mnohem více.

Pozn.: znak <ESC> (ESCAPE) dostaneme, stiskneme-li na počítači 2× za sebou klávesu ESC.

Závěrem bych chtěl poznamenat, že pokud je mi známo, nebyl dosud nikde zveřejněn podrobný popis komunikace počítačů ATARI s periferiemi. Zde zveřejněný popis je výsledkem experimentální práce, analýzy přenosu pomocí datového analyzátoru a simulátoru a disasemblování příslušných rutin operačního systému. Proto nejsou pravděpodobně ošetřeny úplně všechny možné situace, které se při přenosu mohou teoreticky vyskytnout a které nebylo možno nasimulovat. Ze stejného důvodu jsou interfejsem ignorovány zabezpečovací součtové bajty. Pro toto řešení však mluví praxe: Za zhruba půl roku provozu interfejsu nebyla zaregistrována ani jedna chyba při přenosu i při vysoké přenosové rychlosti, která je počítači ATARI používána.

## Literatura

- [1] Eichler, Grohmann: ATARI INTERN, DATA BECKER 1984.
- [2] Intel Component Data Catalog, Intel Corporation 1981.
- [3] Hyan, J., T.: RS 232C – V. 24 AR A10/84.

## Seznam součástek

Mikroprocesor	MHB8048, MHB8035, MHB8748, 18039 nebo ekvivalenty
RAM	2× MHB2114, HM6116 nebo jiná statická RAM
EPROM	MHB8708C, I2716, nebo ekvivalenty
Latch	MHB8282, SN74LS373 nebo jiný osmibitový paralelní střadač
Hradla	MH7400

Tab. 1. Přehled řídicích posloupností pro interfejs (901–T1)

Řídicí posloupnost	Hexa-decimálně	Nastavení interfejsu
<ESC>r CTRL -.	1BH,72H,00H	paralelní výstup z interfejsu (CENTRONICS)
<ESC>r CTRL -A	1BH,72H,01H	sériový výstup z interfejsu 50 Bd
<ESC>r CTRL -B	1BH,72H,02H	" 100 Bd
<ESC>r CTRL -C	1BH,72H,03H	" 200 Bd
<ESC>r CTRL -D	1BH,72H,04H	" 300 Bd
<ESC>r CTRL -E	1BH,72H,05H	" 600 Bd
<ESC>r CTRL -F	1BH,72H,06H	" 1200 Bd
<ESC>j	1BH,6AH	nastavení nejvyššího bitu přenášejících znaků do log.0
<ESC>i	1BH,69H	transparentní přenos nejvyššího bitu
<ESC>d<ESC>c	1BH,64H 1BH,63H	transparentní přenos všech znaků
<ESC>d<ESC>g	1BH,64H 1BH,67H	všechny znaky s kódem nižším než 20H interpretovány jako grafické včetně<ESC>
<ESC>n<ESC>c	1BH,6EH 1BH,63H	všechny znaky s kódem nižším než 20H nahrazovány vlnovkou (ASCII kód 7EH)
<ESC>n<ESC>g	1BH,6EH 1BH,67H	všechny znaky s kódem nižším než 20H interpretovány jako grafické, <ESC> přenášén transparentně
<ESC>z	1BH,7AH	přepnutí režimu interfejsu náhodným příjmem posloupnosti znaků stejné jako některá řídicí posloupnost

# INTERFEJS PRO MAGNETOFON K ATARI

Ing. Milan Kuchař, 739 44 Brušperk 932

V AR A8/87 byl uveřejněn popis interfejsu k magnetofonu pro ATARI 800XL, který byl téměř přesnou kopií zapojení speciálního firemního magnetofonu XC12. Vzhledem k napájecímu napětí 5 V zde není možné bez problémů použít jakékoliv běžné dostupné operační zesilovače. Např. typy 741, 748, 1458 mají nejnižší úroveň na výstupu v nesymetrickém zapojení asi 2 V, což je pro dané zapojení nevyhovující. Z tuzemských OZ je proto vhodné použít např. typy MAB355,6,7 nebo MAC155,6,7. Dále je zapojení poměrně citlivé na výběr součástek v pásmových propustích. Proto jsem vyvinul zapojení interfejsu pracující na jiném principu – s fázovým závěsem.

## Popis zapojení

Záznam informace na kazety u systému ATARI je prováděn tak, že logická „1“ je zaznamenávána kmitočtem 5135 Hz, logická „0“ kmitočtem 3995 Hz a data jsou vysílána rychlostí 600 Bd v sérii s jedním nulovým start-bitem a jedním jedničkovým stop-bitem bez parity. Zaváděcí kmitočet je 5135 Hz – tedy logická „1“. Zapojení interfejsu je na obr. 1. IO1a předzesiluje signál z magnetofonu, komparátor IO1b převádí signál na obdélníky. Z jeho výstupu je signál veden přes C3 na vstup fázového komparátoru PLL (14) (obvod 4046 kondenzátorovou vazbu povoluje). Výstup fázového komparátoru II (13) je veden přes filtr (R9,R10,C5,C6) na vstup VCO (9) a dále z vnitřního emitového sledovače obvodu 4046 (10) přes dolní propust (T14,R15,C7,C8) na T1, který funguje jako transformátor impedance. Z emitoru T1 je signál veden na napěťový kompara-

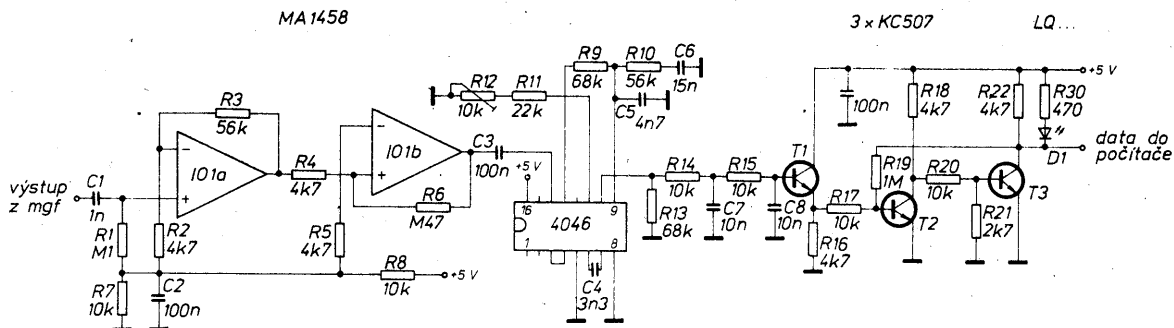
tor s hysterezí – T2,T3, na jehož výstupu je hotový signál pro počítač. Popis funkce fázového závěsu neuvádím, zájemci jej mohou najít např. v [2].

Signál z počítače do magnetofonu je nutno zeslabit a omezit vyšší harmonické kmitočty – viz obr. 2. Trimmerem R24 nastavujeme vstupní napětí pro použitý druh magnetofonu.

## Postup při ožiování

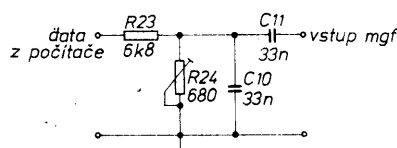
Při použití ověřených součástek interfejs pracuje na první zapojení. Svítivá dioda D1 rozsvícením signalizuje logickou „0“ a zhasnutím logickou „1“.

Seřízení provedeme tak, že nejdříve do interfejsu pustíme z magnetofonu zaváděcí signál některého programu (je před každým



Obr. 1. Schéma zapojení interfejsu (907-1)

programem a trvá 20 s) a trimrem R12 otáčíme tak, aby se svítivá dioda D1 právě rozsvítila; tuto polohu trimru si označíme. Pak pustíme do interfejsu z magnetofonu program a trimrem R12 otáčíme na druhou stranu, až D1 právě přestane blikat. Poté nastavíme R12 doprostřed mezi tyto dvě polohy a seřizování je ukončeno. Jestliže se nám nepodaří najít tyto dvě polohy trimru R12, je nutno změnit odpor rezistoru R11 nebo kapacitu kondenzátoru C4.



Obr. 2. Úprava signálu pro magnetofon (907-2)

### Poznámky ke konstrukci

Návrh plošných spojů neuvádím, protože zařízení je tak jednoduché a malé, že je možno je umístit přímo do magnetofonu. Trimr R12 je vhodně umístěn přístupně – je jím možno korigovat rozdíl otáček při nahrávání programů pořízených na jiném magnetofonu.

### Seznam součástek

Rezistory TR 151:  
R1 100 kΩ  
R2, R4, R5, R16, 4,7 kΩ  
R3, R10 56 kΩ  
R6 470 kΩ  
R7, R8, R14, R15, 10 kΩ  
R17, R20 10 kΩ

R9, R13 68 kΩ  
R11 22 kΩ  
R19 1 MΩ  
R21 2,7 kΩ  
R23 6,8 kΩ  
R30 470 Ω

Potenciometrické trimry:  
R12 10 kΩ  
R24 680 Ω

Kondenzátory keramické:  
C1 1 nF  
C2, C3, C9 100 nF  
C4 3,3 nF  
C5 4,7 nF  
C6 15 nF  
C7, C8 10 nF  
C10, C11 33 nF

Polovodičové prvky:  
IO1 MA1458  
IO2 MHB4046  
T1, T2, T3 KC507 až KC509  
D1 libovolná svítivá dioda

## INTERFEJS ATARI 800 – ALFIGRAF

Ing. Pavel Rada, Mezihorská 56, 140 00 Praha 4

Krokové reverzační motory typu SMR 300-100/24, případně SMR 300-300/24 umožňují stavbu relativně jednoduchých a přitom spolehlivých periferních zařízení k počítačům. Jednou z atraktivních aplikací jsou zapisovače, umožňující jak tisk, tak kresbu grafiky.

Příkladem může být jak tovární konstrukce zapisovače MINIGRAF 0507, tak zapisovač ALFI sestavený ze stavebnice Merkur podle návrhu ing. Dovala.

Zapojení interfejsů k těmto typům zapisovačů bylo zatím publikováno především pro připojení na počítače kompatibilní se ZX Spectrum, kde se využívá obvodu MHB 8255.

Osmibitové Atari mají již obvod vykonávající základní funkce interfejsu zabudovaný uvnitř (PIA) a vyvedený na dva ovladačové konektory. To umožňuje stavět značně jednodušší konstrukce. Například pákové ovladače pro Atari obsahují jen spínací kontakty.

Programově lze přes ovladačové kontakty vyvést i osmibitovou sběrnici a umožnit tak snadné a jednotné připojení periférií k celé typové řadě osmibitových Atari.

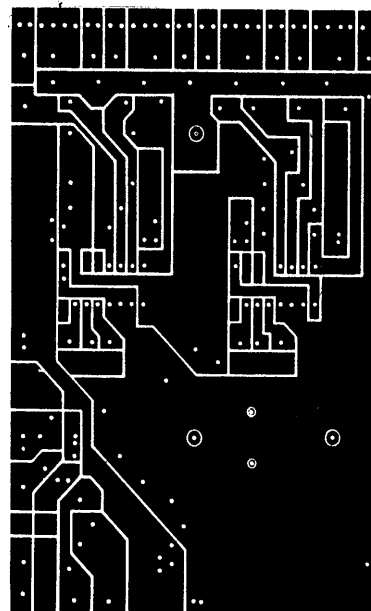
Konstrukční řešení navazující na uspořádání zapisovače ALFI a využívající programového vybavení ing. Jandíka pro MINIGRAF bylo nazváno ALFIGRAF. Jeho zapojení je na obr. 1.

Soubor programů v systému Turbo 2000 pro zapisovače typu ALFIGRAF nebo Minigraf, včetně textového editoru, byl sestaven na magnetofonovou kazetu typu C60, kde zaplňuje téměř celou jednu stranu a obsahuje souhrně přes 2 x 150 kB dat.

Zápis na dodanou kazetu zajišťuje i nečlenům klubu, proti proplacení složenky (100 Kčs), S.W. skupina klubu, a lze jej objednat zaslaním korespondenčního listu s označením SYKO 007/89 na adresu:

487. ZO SVAZARMU ATARI KLUB PRAHA, pošt. příhrádka 51, 100 00 Praha 10.

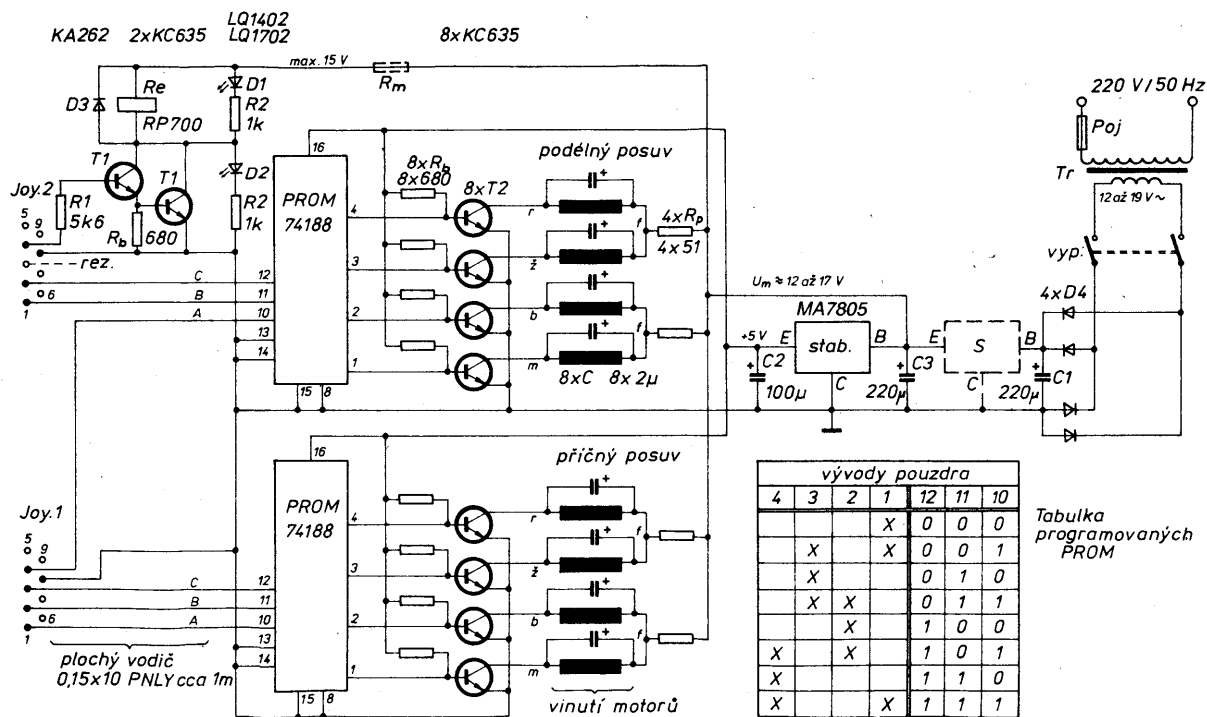
Pro uvedené zapojení je nezbytné použít naprogramovanou bipolární paměť PROM MH74188. Způsob naprogramování, při kterém je využita jen malá část kapacity paměti, je uveden na obr. 2, kde křížky označují logickou hodnotu 1 při zapojení otevřených kolektorů přes rezistor 680 Ω na plus 5 V.



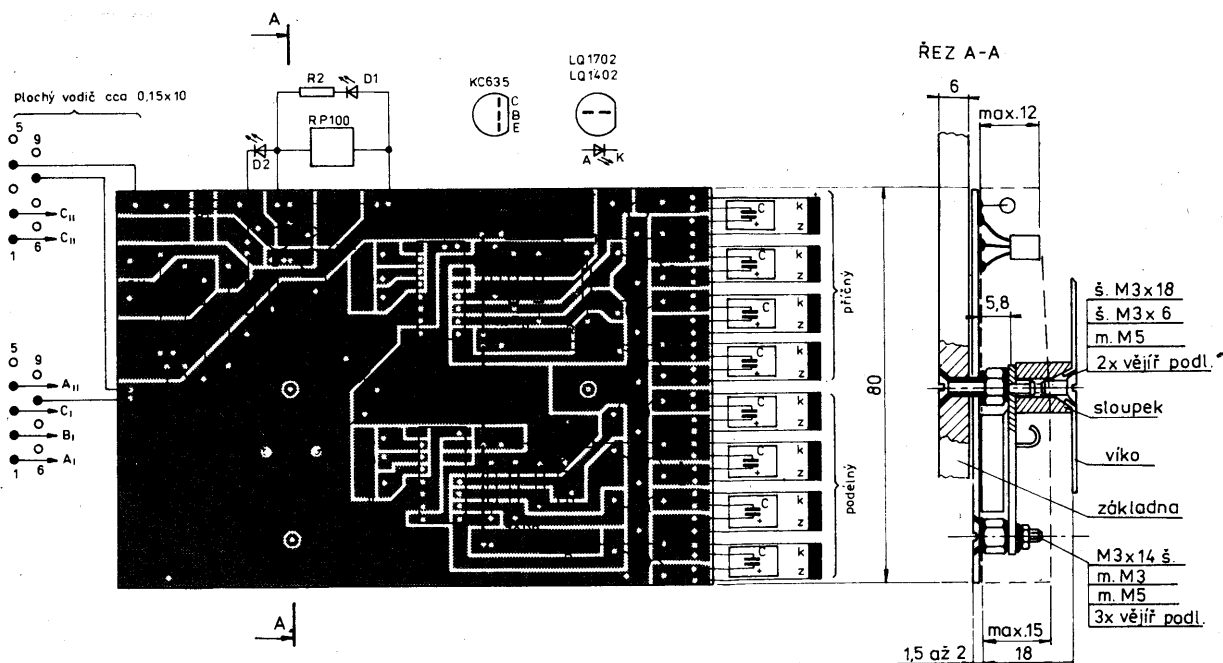
Obr. 2. Obrázek plošných spojů desky Y505 (906-2)

Zapojení tohoto typu je odolné proti změně napájecích napětí a variabilitě součástek. Po montáži proto postačí pečlivá vizuální kontrola správnosti zapojení konektorů a obvodů na desce s plošnými spoji a při ožiování postačí měřicí přístroj typu Avomet.

Deska na obr. 3 je jednostranná a plošné spoje jsou vytvořeny dělicími čarami. Součástky jsou pájeny ze strany spojů. Odpadá tak vrtání otvorů, zvýší se přehlednost a variabilita pro použití různých náhrad za zá-



Obr. 1. Schéma Alfigrafu pro Atari 800XL/XE, 130XE (906-1)



Obr. 3. Rozmístění součástek na desce Y505 a celkové konstrukční uspořádání. Všechny součástky se pájejí ze strany mědi. Vývody PROM č. 5, 6, 7 a 9 nezapojujte! Odehněte je do vzduchu! (906-3)

kladní typy součástek. Velké plochy mědi pomáhají odvádět teplo, kterého se uvolňuje nejvíce z rezistorů i při klidovém stavu motorů.

Rozměrově je deska určena k zabudování do ALFIGRAFU, ale při zajištění odvodu tepla ze stabilizátoru jiným způsobem než přes sloupek na víko kryjící elektroniku lze zapojení použít i pro jiná konstrukční řešení.

### Použité podklady

Návod ze stavebnic Merkur – ALFI.  
Návod k obsluze MINIGRAFU a jeho modulu pro Atari.  
Zpravodaje 487. ZO Svazarmu č. 1; 4; 6/1989.

### Použité součástky

S	pro $U_{sek} > 17$ (14) V použít stabilizátor MA7812(7815)
T1	KC635 $h_{21E} > 160$ nebo Darlington
T2	KC635 (KC637, K4639)
C	2 $\mu$ F/70 V TE006
C1, C3	220 $\mu$ F/40 V TF010
C2	100 $\mu$ F/6 V TE 981

R1	4,7 k $\Omega$ i více podle zesílení T1
R2	1 k $\Omega$ (v sérii s D1 může být jen 510 $\Omega$ )
R <sub>b</sub>	680 $\Omega$ i více
R <sub>m</sub>	předřadný odpor, použit pro $U_m > 15$ V
R <sub>p</sub>	33 až 51 $\Omega$ /2 – 4 W volit dle $U_m$
D1	LQ1402 žlutá
D2	LQ1702 zelená
D3	KA262 nebo KY13 ...
D4	KY132/80
Tr	transformátor min. 20 W
Motory	SMR 300-100 RI/24 (r – rudá, f – fialová, b – bílá, m – modrá, z – žlutá)

# AUTOMATIZOVANÝ EXPOZIČNÝ SYSTÉM

Ing. Ivan Hejda, Jašíkova 15, 821 03 Bratislava

Ing. Ján Kačmárík, nám. Lud. milície 10, 821 09 Bratislava

Nasledujúce zapojenie spolu s uvedeným programovým vybavením značne zľahčuje práce vo fotokomore pri čiernobieliom pozitívnom procese. Je pripojené na mikropočítač ZX-Spectrum cez paralelný interfejs s MHB8255 (AR 6/85), plne nahradzuje osvitomer a spínacie hodiny. Na základe nameraného jasu snímky vypočítava expozičnú dobu, meria teplotu vyvolávacieho kúpeľa, resp. prepieracej vody. Všetky operácie vrátane spínania zväčšovacieho prístroja vykonáva užívateľ prostredníctvom klávesnice mikropočítača a mikrosplínača. Spoluprácu s obslužným programom zľahčujú pomocné údaje na obrazovke, ktorá je kvôli zatemneniu prikrytá červenou fóliou. Užívateľ má možnosť použiť vypočítanú dobu expozície, prípadne ju opraviť podľa vlastného uváženia a opakovať expozíciu pri opakovanom zväčšovaní tej istej snímky.

Podstatnú časť zapojenia tvoria snímač intenzity osvetlenia a výkonový spínač.

Použitie zariadenia je programovým vybavením viazané na mikropočítač ZX-Spectrum. Po prepísaní programu je použiteľné s ľubovoľným mikropočítačom, ktorý je vybavený paralelným interfejsom.

Hlavné použité súčiastky: E555, WK65061, 11NR15, MHB4011, impulzný transformátor.

## Popis konštrukcie

- Schéma zapojenia je na obr. 1, obrazec plošných spojov na obr. 2 a zapojenie konektorov k paralelnému interfejsu a snímačom na obr. 4.

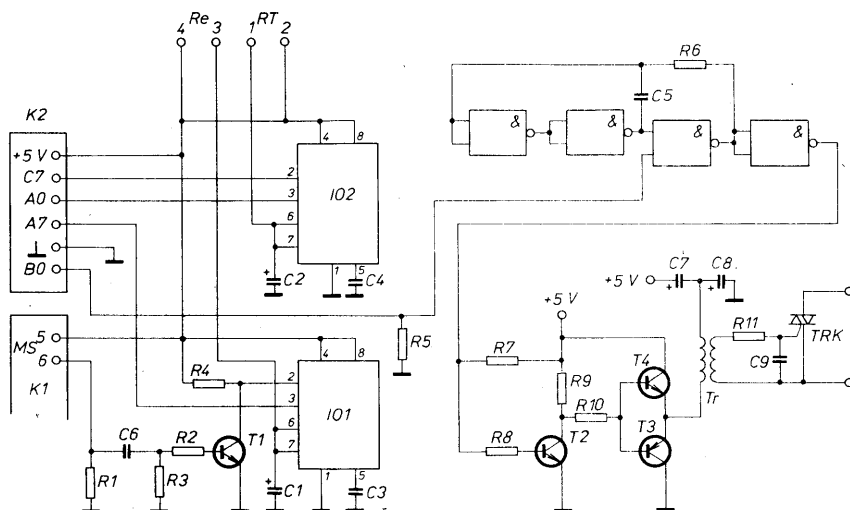
Výkonový spínač pre ovládanie zväčšovacieho prístroja je riadený bitom B0 MHB8255. Časť merania teploty vývojky resp. prepieracej kúpele s IO E555 je identická s prevodníkom (AR 6/88). Štartovací impulz prevodu je generovaný mikropočítačom cez bit C7 MHB8255, výstupný impulz prevodníka je pripojený na bit A0. Termistor 11NR15 je cez ohybný vodič pripojený k bodom 1,2 zapojenia, aby sa dal umiestniť priamo do média, teplotu ktorého meriame.

Snímač jas snímky IO1 pracuje na tom istom princípe s tým, že štartovací impulz prevodu nedáva mikropočítač, ale obsluha stlačením mikrosplínača pripojeného k bodom 5,6 zapojenia. Samotný fotorezistor WK 65061 je cez ohybný vodič pripojený k bodom 3,4. Je umiestnený spolu s mikrosplínačom na jednoduchom držadle (celok v ďalšom texte nazývame sondou). Obsluha takto môže umiestniť fotoodpor do želaného miesta exponovanej snímky (ako pri osvitomere) a mikrosplínačom odštartovať prevod. U oboch snímačov je prevod 16-bitový s ochranou proti preplneniu počítadla.

## Popis obsluhy

Ovládanie AES zabezpečuje program v jazyku BASIC (Výpis 1) využívajúci spomínané snímače pomocou podprogramu v strojovom kóde (Výpis 2). Výkonový spínač je riadený inštrukciou OUT.

Po spustení programu má užívateľovi 5 možností, ktoré sa volia stlačením týchto klávesov:



Obr. 1. Schéma zapojenia (911-1)

- "t" – expozičný test: tento je treba vykonať, ak chce užívateľ používať automatické určovanie doby expozície (zmienime sa o ňom neskôr),
- "k" – zmena expozičnej konštanty: túto užívateľ vykoná ak sa zmenia podmienky vyvolávania resp. podľa svojho uváženia (o tejto možnosti sa tiež zmienime neskôr).
- "e" – skok do expozičného bloku, ktorý je jadrom práce s AES.
- "f" – meranie teploty vývojky resp. prepieracieho kúpeľa. Automaticky se zmeria a zobrazí príslušná teplota s presnosťou na 0,1 °C.
- "c" – podprogram na ciachovanie termistora. Toto ciachovanie vykonáme hneď pri prvom použití AES, pričom namerané konštanty sa uložia na pásku za samotný obslužný program. Pri ďalšom používaní už netreba teplomer ciachovať, príslušné konštanty sa nahrávajú z pásky.

Po voľbe expozičného bloku sa zobrazí "menu" ponúkajúce tieto možnosti:

- "Z" – zadanie požadovanej doby expozície v sekundách z klávesnice.
- "B" – meranie bodovým snímačom. Pri zvolení tejto činnosti sa predpokla-

dá, že v ráme zväčšovacieho prístroja je pripravený negatív snímky. Automaticky sa zapne zväčšovací prístroj. Užívateľ si vyberie ľubovoľný bod snímky a zadá cez klávesnicu aký stupeň sčernania má mať pozitív v tomto mieste. Stupeň sčernania sa volí podľa stupnice šedi, o ktorej sa zmienime neskôr. Potom položí sondu na zvolené miesto a mikrosplínačom odštartuje meranie. Po skončení prevodu sa automaticky vypne zväčšovací prístroj, vypočíta a zobrazí sa doba expozície.

- "E" – expozícia. Do zväčšovacieho rámu sa založí fotocitlivý papier. Po stlačení klávesy "e" sa zapne zväčšovací prístroj na dobu, ktorá zodpovedá naposledy zadanej alebo vypočítanej dobe expozície. Opätovným stlačením klávesy možno zopakovať expozíciu toho istého snímku niekoľko krát.

- "1/0" – zapnutie a vypnutie zväčšovacieho prístroja prostredníctvom klávesnice. Túto možnosť volí užívateľ pri zaostrovani snímku, hľadani výrezu, atď.

Teraz sa vráťme k expozičnému testu. Je nevyhnutný, lebo do určovania doby expozície treba zahrnúť aj vlastnosti vývojky ako i emulzie používaného fotopapiera. K expo-





```

1054 PRINT "C - Ciachovanie te
mistora"
1060 PAUSE 0
1070 BEEP 0.02,12
1100 IF INKEY$="e" THEN GO TO 20
00
1110 IF INKEY$="t" THEN GO TO 15
00
1130 IF INKEY$="k" THEN GO TO 17
00
1132 IF INKEY$="f" THEN GO TO 30
00
1134 IF INKEY$="c" THEN GO TO 3
500
1140 GO TO 1060
1230 PAUSE 0
1490 REM Expozicny test
1500 CLS : PRINT "Expozicny test"
1505 INPUT "Optimalna doba expoz
icie testu- vacieho snimku ";ttest
1510 PRINT AT 10,2;"Zosnimaj ska
lu sedi bodovym sn
imacom"
1520 LET pp=10: REM 10 stupnov
sedi
1530 DIM e(pp)
1540 OUT 63,1: REM Zapnutie zvec
sovacieho pristroja
1550 FOR i=1 TO pp
1560 PRINT AT 13,12,i;" stlpec"
1570 LET bc=USR 50000: PRINT AT
21,25,BC;
1580 IF bc=0 THEN GO TO 1500
1590 GO SUB 2700: REM Kvadratick
a aproximacia
1600 LET e(i)=jas
1610 NEXT i
1620 GO TO 1000
1630 REM Korekcia expozicnej
doby
1700 CLS : PRINT "Korekcia expoz
icnej doby"
1710 INPUT "Modifikovana opt. do
ba expozicieskusobneho snimku ";
ttest
1720 GO TO 1000
2000 REM Expozicny blok
2010 REM Menu
2020 CLS : PRINT " E X P O Z I C
I O
N E
M E N U"
2030 PRINT "Z - zadanie doby e
xpozicie"
2040 PRINT "B - meranie bodovy
m snimacom"
2050 PRINT "T - expozicia (Tex
=";ex;"")
2070 PRINT "1 - zapnutie zvets
. pristroja"
2080 PRINT "0 - vypnutie zvets
. pristroja"
2090 PRINT "M - navrat do menu"
2100 PAUSE 0
2105 BEEP 0.02,12
2110 IF INKEY$="z" THEN GO TO 22
00
2120 IF INKEY$="b" THEN GO TO 25
00
2130 IF INKEY$="g" THEN GO TO 26
00
2140 IF INKEY$="1" THEN OUT 63,1
: GO TO 2100
2150 IF INKEY$="0" THEN OUT 63,0
: GO TO 2100
2170 IF INKEY$="e" THEN GO TO 23
00
2180 IF INKEY$="m" THEN GO TO 10
00
2190 GO TO 2100
2199 REM Zadanie doby expozicie
2200 CLS : PRINT "Zadavanie doby
expozicie"
2210 INPUT "Tex=";ex
2220 GO TO 2000
2300 REM Expozicia
2305 IF ex=0 THEN GO TO 2000
2310 CLS : PRINT "Expozicia"
2320 POKE 23673,0: POKE 23672,0
2330 OUT 63,1
2360 LET ex1=(256*PEEK 23673+PEE
K 23672)/50
2365 IF ex-ex1<0 THEN GO TO 2390
2370 PRINT AT 10,12;ex-ex1;"
2380 GO TO 2360
2390 OUT 63,0
2400 GO TO 2000
2499 REM Meranie bodovym
snimacom
2500 CLS : PRINT "Meranie bodovy
m snimacom"
2510 INPUT "Zadaj ziadany stupeň
scernania meraneho miesta ";ss
C
2520 PRINT AT 10,3;"Zvol merane
miesto, aktivuj sn
imac"
2530 OUT 63,1: REM Zapnutie zvec
sovacieho pristroja
2540 LET bc=USR 50000
2545 OUT 63,0: REM Vypnutie zvec
sovacieho pristroja
2550 IF bc=0 THEN CLS : PRINT FL
ASH 1:AT 10,1;"Intenzita mimo ro
zsa hu snimaca": PAUSE 0: BEEP .1
,12: GO TO 2000
2560 GO SUB 2700
2570 LET ex=e(ssc)/jas*ttest
2580 GO TO 2000
2700 REM Kvadraticka aproximacia
prevedovej charakteristiky
2710 IF bc<730 THEN LET in=1: GO
TO 2770
2720 IF bc<1810 THEN LET in=2: G
O TO 2770
2730 IF bc<4290 THEN LET in=3: G
O TO 2770

```

```

2740 IF bc<11380 THEN LET in=4:
GO TO 2770
2750 IF bc<29800 THEN LET in=5:
GO TO 2770
2760 LET in=6
2770 LET jas=a(in)*bc*bc+b(in)*b
c+(in)
2780 RETURN
3000 REM Podprogram pre meranie
teploty
3005 LET as="Aktivuj snimac tepl
oty"
3010 CLS : PRINT AT 16,5; FLASH
1,3
3020 PRINT AT 20,5;"M - navrat d
o menu"
3030 IF INKEY$="m" THEN GO TO 10
00
3040 LET ts=USR 50026
3050 IF ts=0 THEN GO TO 3010
3055 GO SUB 3100: REM Zaradenie
do i-teho intervalu merania.
3060 LET tc=m(2,i)*ts+2+m(3,i)*t
s+m(4,i)
3070 PRINT AT 16,5;"
3080 PRINT AT 10,8;"Teplota=";tc
;"C"
3090 GO TO 3030
3100 IF ts<=m(1,1) OR ts>=m(1,21)
THEN PRINT AT 0,0; FLASH 1;"Tep
ota prekrocila ciachovany
rozsa h": GO TO 3120
3110 PRINT AT 0,2;"
3120 IF ts>=m(1,11) THEN GO TO 31
80
3130 IF ts>=m(1,7) THEN GO TO 317
0
3140 IF ts>=m(1,1) AND ts<=m(1,3
) THEN LET i=1: RETURN
3150 IF ts>=m(1,3) AND ts<=m(1,5)
THEN LET i=2: RETURN
3160 IF ts>=m(1,5) AND ts<=m(1,7)
THEN LET i=3: RETURN
3170 IF ts>=m(1,7) AND ts<=m(1,9)
THEN LET i=4: RETURN
3180 IF ts>=m(1,9) AND ts<=m(1,11
) THEN LET i=5: RETURN
3190 IF ts>=m(1,15) THEN GO TO 32
20
3200 IF ts>=m(1,11) AND ts<=m(1,
13) THEN LET i=6: RETURN
3210 IF ts>=m(1,13) AND ts<=m(1,1
5) THEN LET i=7: RETURN
3220 IF ts>=m(1,15) AND ts<=m(1,1
7) THEN LET i=8: RETURN
3230 IF ts>=m(1,17) AND ts<=m(1,1
9) THEN LET i=9: RETURN
3240 LET i=10: RETURN
3500 REM Podprogram pre ciachov
anie termistora
3502 CLS
3505 PRINT AT 7,6; INVERSE 1;"Ci
achovanie termistora"
3510 FOR i=1 TO 21: PRINT AT 10,
7;"Meranie cislo";i: INPUT "Tep
lota=";y(i)
3520 LET m(1,i)=USR 50026
3530 NEXT i
3540 CLS : PRINT AT 10,2; FLASH
1;"Prebieha vypocet aproximacie"
3550 FOR i=1 TO 10
3560 LET p=2*i-1: LET d=2*i: LET
t=2*i+1

```

```

3570 GO SUB 3600: REM Riesenie
sustavy lin. rovnici Cramerovym
pravidiom.
3580 NEXT i
3590 SAVE "m" DATA m$( )
3595 GO TO 1000
3600 LET d=m(1,d)*m(1,p)+2+m(1,t
)*m(1,d)+2+m(1,p)*m(1,t)+2-m(1,d
)*m(1,t)+2-m(1,t)*m(1,p)+2-m(1,p
)*m(1,d)+2
3610 LET d1=y(d)*m(1,p)+2+y(t)*m
(1,d)+2+y(p)*m(1,t)+2-y(d)*m(1,t
)+2-y(t)*m(1,p)+2-y(p)*m(1,d)+2
3615 LET d2=y(d)*m(1,p)+2+y(t)*m
(1,d)+2+y(p)*m(1,t)+2-y(d)*m(1,t
)+2-y(t)*m(1,p)+2-y(p)*m(1,d)+2
3620 LET d3=y(t)*m(1,d)*m(1,p)+2
+y(d)*m(1,p)*m(1,t)+2+y(p)*m(1,t
)*m(1,d)+2-y(p)*m(1,d)*m(1,t)+2-
y(d)*m(1,t)*m(1,p)+2-y(t)*m(1,p
)*m(1,d)+2
3630 LET m(2,i)=d1/d
3640 LET m(3,i)=d2/d
3650 LET m(4,i)=d3/d
3660 RETURN

```

## Výpis 2. Rutina pre obsluhu snímačov a pre- vodníka (911-V2)

```

10 ; Rutina pre obsluhu
20 ; snímačov a prevodníka.
30
40 ORG 50000
45
50 DI
55 LD BC,#0000
60 LC354 IN A,(#1F)
65 ALA
70 JR NC,LC354
75 LC359 INC BC
80 INC B
85 JR Z,LC365
90 DEC B
95 IN A,(#1F)
100 ALA
105 JR C,LC359
110 EI
115 RET
120 LC365 LD BC,#0000
125 EI
130 RET
135 DI
140 LD BC,#0000
145 LD A,#00
150 OUT A,(#5F),A
155 LD A,#50
160 OUT A,(#5F),A
165 LC376 INC BC
170 INC B
175 JR Z,LC382
180 DEC B
185 IN A,(#1F)
190 RRA
195 JR C,LC376
200 EI
205 RET
210 LC382 LD BC,#0000
215 EI
220 RET

```

# MINIGRAF - ZX SPECTRUM

Ing. M. Šedivý, Hradešinská 25, 101 00 Praha 10

Článek popisuje využití souřadnicového zapisovače MINIGRAF ve spojení s počítačem ZX Spectrum. Minigraf je výstupní grafická jednotka, pracující v soustavě pravoúhlých souřadnic nezávisle ovládaných vnějším signálem. Použitím MINIGRAFu získáme periférii, vhodnou např. ke kreslení grafů, tabulek, případně obrázků. MINIGRAF však nenahrazuje tiskárnu, především z důvodu malé rychlosti kreslení. Při současném nedostatku podobných zařízení na našem trhu může přínést použití MINIGRAFu určité oživení vašeho malého výpočetního systému.

Souřadnicový zapisovač MINIGRAF kreslí v pravoúhlých souřadnicích na běžný kancelářský papír formátu A4, případně na svítek maximální šíře 210 mm. Pisátko je kuličkový fix nebo jiné běžné dostupné pero. Posuv písátka ve směru osy X a posuv papíru ve směru osy Y zajišťují krokové motorky. Maximální rychlost kreslení je 80 mm/s, délka kroku je 0,125 mm. MINIGRAF

je ovládán přes jednoduché jednosměrné rozhraní. Dvěma trojicemi vodičů, označených X1, X2, X4 a Y1, Y2, Y4 jsou ovládány fáze krokových motorků. Sedmý vodič ovládá spouštění písátka. Pohyb motorků a směr pohybu je řízen rostoucí nebo klesající binární posloupností příslušné trojice bitů. MINIGRAF nedává žádné zpětné hlášení, proto je nutno zajistit časové prodlevy mezi změnami

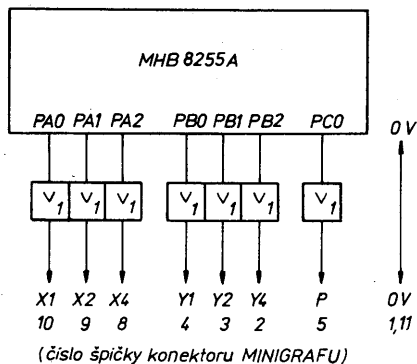


KROK	VSTUP		
	X4(Y4)	X2(Y2)	X1(Y1)
0	L	L	L
1	L	L	H
2	L	H	L
3	L	H	H
4	H	L	L
5	H	L	H
6	H	H	L
7	H	H	H
8	L	L	L
9	L	L	H
10	L	H	L
11	L	H	H
12	H	L	L
13	H	L	H
14	H	H	L
15	H	H	H
16	L	L	L

Tab. 1. Posloupnost kroků (904–T1)

nami binární kombinace programově. Posloupnost jednotlivých kroků je patrná z tabulky č. 1. Při rostoucí posloupnosti (např. kroky 7,8,9) se pohybuje písátka i papír v kladném směru os (tj. zleva doprava a zezadu dopředu). Jeden krok znamená posun papíru nebo písátka o 0,125 mm. Maximální kmitočet krokování je 400 Hz. Při reverzaci je nutná prodleva 10 ms a po sklopení nebo zdvižení písátka je nutná prodleva 50 ms. Písátka je spouštěno úrovní log.1.

Z uvedeného vyplývá, že pro připojení MINIGRAFu k počítači musíme použít jednu osmibitovou bránu (jeden bit zůstane nevyu-



Obr. 1. Propojení interfejsu s Minigrafem (na místě oddělovacích zesilovačů lze použít např. 2x 7407, 1x 3212, 2x 3216, 1x 8282 ap.) (904–1)

žit). Pro připojení MINIGRAFu k počítači ZX Spectrum můžeme s výhodou použít interfejs s MHB8255A podle AR č. 6, 1985, s. 217.

Při propojení jsem pro jednodušší programové řešení využil všechny tři brány obvodu 8255. Každý ovládací bit má v MINIGRAFu udávanou log. zátěž 5, proto je vhodné všechny signály zesílit. Propojení interfejsu s MINIGRAFeM je na obr. 1. Po propojení a zadání následujícího jednoduchého programu by se měl motor pro posun papíru točit.

```
10 OUT127,128
20 FOR A=0 TO 15
30 OUT 63,A
40 NEXT A
50 GOTO 20
```

Zaměníme-li výstup 63 za 31, měl by se točit motor pro posun písátka. Instrukci OUT 95, 255 spustíme písátka. Tím je propojení odzkoušeno a můžeme přistoupit k programovému řešení.

Ovládání MINIGRAFu v jazyce BASIC je pomalé, a proto je vhodné psát programy ve strojovém kódu. Uvádím příklad programu, který provádí kopii obrazovky na souřadnicovém zapisovači. Program je uložen na konci paměti za RAMTOP. Program nahrajeme např. takto: CLEAR 65043 : LOAD " " CODE 65044 : POKE 65372, X : POKE 65414, X. Proměnná X může nabývat hodnot od 1 do 8 a určuje rozměr kopie. Po nahrání je nastaveno X = 4. Program spustíme příkazem RANDOMIZE USR 65044. Po spuštění programu nastavíme ručně písátka do výchozího bodu při levém okraji na začátku papíru. K ručnímu ovládání posuvu písátka a papíru slouží tlačítka se šipkami pro ovládání kurzoru.

Kreslení spustíme tlačítkem EDIT, přerušeni kreslení a návrat z programu způsobí stisk tlačítka CAPS LOCK. Návrat před spuštěním kreslení je možný tlačítkem DELETE. Po ukončení kresby se automaticky provede návrat z programu.

Pro výpis programu není pro svou pomalost zapisovač vhodný, ale v nouzi poslouží také. Pro kreslení je naopak velmi vhodný, umožňuje jemnější kresbu než obrazovka nebo tiskárna.

## Literatura

- 1 | Soldán, J.: Interfejs s MHB8255. AR/A, č. 6, 1985, s. 217–219.
- 2 | MINIGRAF 0507, návod k obsluze, Aritma 1986.

## Výpis 1. Strojový kód programu Zapisovač (904–V1)

```
65044: 243 245 213 62 128 211 127 17 0 0
65054: 62 0 211 95 62 239 219 254 203 103
65064: 194 44 254 29 203 95 194 50 254 28
65074: 203 87 194 56 254 20 203 71 202 87
65084: 254 62 247 219 254 203 103 194 71 254
65094: 21 203 71 202 84 254 217 205 96 254
65104: 217 195 34 254 205 156 254 209 241 251
65114: 201 0 0 0 0 0 217 122 211 31
65124: 123 211 63 205 112 254 217 201 0 0
65134: 0 0 197 6 2 14 255 13 194 117
65144: 254 5 194 115 254 193 201 205 112 254
65154: 205 112 254 205 112 254 205 112 254 201
65164: 205 127 254 205 127 254 205 127 254 201
65174: 0 0 0 0 0 0 197 213 229 245
65184: 30 3 217 197 229 17 0 0 1 0
65194: 0 217 33 0 64 6 8 14 8 22
65204: 32 217 6 0 217 52 53 196 0 255
65214: 35 217 62 8 128 71 217 21 194 185
65224: 254 197 1 224 0 9 193 217 12 217
65234: 205 127 254 13 194 179 254 197 1 32
65244: 248 9 193 5 194 177 254 197 1 0
65254: 7 9 193 29 194 175 254 217 1 0
65264: 0 217 205 0 255 217 225 193 217 241
65274: 225 209 193 201 0 0 197 213 217 120
65284: 186 212 19 255 210 15 255 21 195 107
65294: 255 20 195 114 255 121 187 202 39 255
65304: 210 31 255 29 195 121 255 28 195 128
65314: 255 0 0 0 0 217 6 8 126 87
65324: 122 23 87 218 77 255 62 0 211 95
65334: 205 140 254 195 200 255 195 160 255 5
65344: 194 44 255 62 0 211 95 205 140 254
65354: 209 193 201 62 255 211 95 205 140 254
65364: 195 57 255 0 07 0 197 6 4 124
65374: 211 63 60 103 205 112 254 5 194 93
65384: 255 193 201 213 17 0 255 195 132 255
65394: 213 17 0 1 195 132 255 213 17 255
65404: 0 0 195 132 255 213 17 1 0 197 6
65414: 4 124 211 63 130 103 125 211 31 131
65424: 111 205 112 254 5 194 135 255 193 209
65434: 195 183 255 0 0 217 205 90 255
65444: 217 195 63 255 0 0 0 0 0 0
65454: 0 0 0 0 0 0 0 0 0 0
65464: 247 219 254 203 79 194 3 255 217 209
65474: 193 51 51 195 245 254 217 20 62 0
76484: 186 194 214 255 22 255 217 195 63 255
65494: 217 195 60 255
```

## PROGRAMOVÁNÍ PAMĚTI EPROM na počítačích ATARI XE/XL

Jiří Pokorný, Jan Pokorný, Jiří Pokorný, Fr. Kadlece 18, 180 00 Praha 8

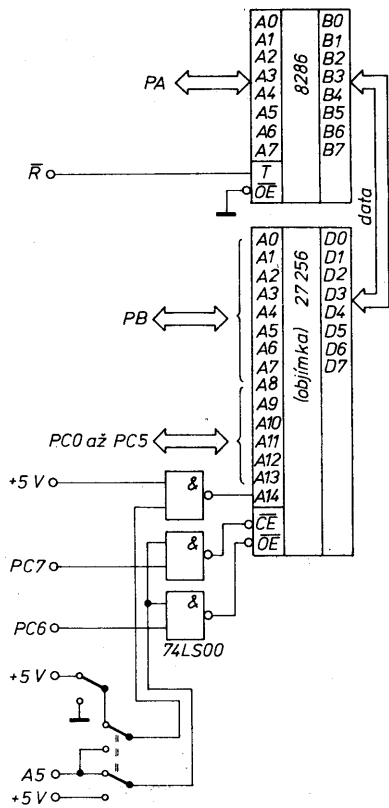
Standardní programátory paměti EPROM nejsou pro většinu amatérů dostupné. Pro malé paměti se nejčastěji amatérsky vyrábějí „ruční“ programátory, pro větší paměti je nutno programovat pomocí řídicího mikropočítače. Uvedeme popis programování paměti EPROM 27128 (16 kB) a 27256 (32 kB) s využitím počítače ATARI řady XL nebo XE. Popsaný postup byl použit při ověření funkce programového modulu (cartridge) s jazyky BASIC XE – MAC/65.

## Technické řešení

Adresy, data i řídicí signály se přivádějí na paměť EPROM přes paralelní interfejs s MHB 8255A, jehož popis a schéma jsou uvedeny v [1]. Paralelní interfejs je zapojen k počítači ATARI v konektoru pro programový modul a na joystickové porty. K paralelnímu interfejsu je připojen přípravek s objemkou pro paměť EPROM. Schéma zapojení přípravku je na obr. 1. Adresy programované EPROM A0 až A7 jsou připojeny na port PB obvodu 8255 A, adresy A8 až A13 na část portu PC. Při programování EPROM 32 kB je přes přepínače a invertor přiváděna na vývod číslo 27 (kde je adresa A14) buď úroveň log.0 (programování v rozmezí \$0 až \$3FFF) nebo úroveň log.1 (programování v rozmezí \$4000 až \$7FFF). Při programování EPROM 16 kB se tento vývod používá jako PGMn. Data jsou vyvedena z portu PA

na oddělovací obvod MHB 8286. Směr průchodu dat tímto obvodem lze ovládat signálem Rn obvodu 8255A nebo signálem OEn programované EPROM. Řídící vstupy EPROM CEn, OEn jsou ovládány bity PC7, PC6. Při programování EPROM 32 kB jsou tyto signály kombinovány s latchingovou adresou A5, která působí jako hradlový signál. Při programování EPROM 16 kB latchingová adresa A5 vytváří programovací impuls pro PGMn. V zapojení nejsou zakresleny příklady napájecích napětí  $U_{CC}$ ,  $U_{PP}$  a příslušné blokové kondenzátory (paralelně 100 nF a 5  $\mu$ F). Provozní hodnoty  $U_{CC}$  i  $U_{PP}$  jsou +5 V. Při programování je  $U_{CC}$  = 6 V a  $U_{PP}$  = 12,5 V. U některých EPROM 27128 je  $U_{PP}$  = 21 V (neuvádíme rozmezí napětí).

Obvod 74LS00 se používá jako invertor a oddělovací stupeň. Při programování EPROM 27256 působí rovněž jako hradlo pro signály řízené CEn a OEn.



Obr. 1. Schéma přípravku k programování EPROM 27256 a 27128 (915-1)

## Programové řešení

Algoritmy programování paměti EPROM včetně vývojových diagramů jsou uvedeny v katalogu firmy Intel. Používáme algoritmus nazývaný „inteligent“. Programuje se milisekundovými impulsy; po každém z nich následuje verifikace zapsaného bajtu. Pokud byl bajt zapsán správně, programuje se závěrečným impulsem („overimpulsem“). Délka tohoto impulsu je trojnásobkem celkové délky všech milisekundových impulsů potřebných k zapsání tohoto bajtu.

Jestliže se programovaný bajt nezapsal ani po 25 impulsích délky 1 ms, je ohlášena chyba. Po zapsání všech bajtů programu do paměti EPROM se provede verifikace celého programovaného úseku při provozních hodnotách napětí.

Hlavní program je vypracován v ASSEMBLERU CPU 6502 a je uložen od adresy \$8000. Je volán obslužným programem v BASIC. Programy jsou uvedeny ve Výpisu 1 a Výpisu 2.

## Ovládání obou programů

### RUN + RETURN

Start programu. Na obrazovce se zobrazí text:

„Programování 16/32 kB EPROM (zadej 16 nebo 32) ...?“

Po zadání požadované hodnoty a stisku klávesy RETURN se na displeji zobrazí text „Adresa EPROM, od které se začíná programovat ...?“.

Na tuto výzvu uložíme příslušnou adresu (decimálně) a stiskneme RETURN. Na displeji uvidíme

„Adresa RAM, kde začínají data pro programování EPROM ...?“.

Zadáme adresu začátku dat (decimálně) a stiskneme RETURN.

Na displeji se zobrazí další text:

„Adresa RAM, kde končí data pro programování ...?“.

Zadáme koncovou adresu RAM, kde jsou uložena data. Program pokračuje po stisku RETURN. Na displeji se zobrazí

„Kontroluj napětí Vcc a Vpp (+5 V) na paměti EPROM! (Čekám na klávesu!)“.

Po stisku libovolné klávesy tento program v BASIC vyvolá program ve strojovém kódu

pro programování EPROM. Na displeji uvidíte

„Nastav napětí Vcc = 6 V, Vpp = 12,5 V. (Čekám na klávesu! – A, B)“.

Na tuto výzvu nastavíte programovací napětí v uvedeném pořadí na paměti EPROM (podle popisu v katalogu Intel). Pro některé EPROM 27128 je třeba  $V_{pp} = 21$  V. Po stisknutí kláves A a B v uvedeném pořadí program provádí potřebné programovací opera-

Výpis 2. Program ve strojovém kódu k programování paměti EPROM (ukládá se od adresy \$8000) (915-V2)

```
68 20 02 B1 AD 11 06 F0 06 20 12 B1 4C 12 80 20 0A B1 20 0521
EB 80 20 72 B1 20 90 B1 20 11 B3 A0 00 B1 CB 20 74 B3 20 07B6
25 B1 A9 00 BD 00 D4 A5 CB BD 0B 06 A5 CC BD 0C 06 A2 00 0770
A0 00 B1 CB BD 0B 06 A0 01 20 9F B1 EB E0 1D 00 03 4C CB 0863
80 20 39 B2 AD 0B 06 CD 09 06 D0 EB 8A BE 0A 06 18 6D 0A 0661
06 6D 0A 06 A8 20 9F B1 EE 06 06 D0 03 EE 07 06 E6 CB D0 07B4
02 E6 CC A5 CB CD 04 06 D0 B1 A5 CC CD 05 06 D0 AA AD 2F 0A1B
02 BD 00 D4 20 1A B1 A9 00 BD 00 D4 20 EB 80 0A 00 20 C3 0736
82 A0 00 B1 CB CD 09 06 D0 37 E6 CB D0 02 E6 CC EE 06 06 09B0
D0 03 EE 07 06 A5 CB CD 04 06 D0 DF A5 CC CD 05 06 D0 D8 09B5
AD 2F 02 BD 00 D4 20 84 B1 20 3B B1 60 20 39 B2 AD 0B 06 0636
CD 09 06 D0 03 4C 58 80 20 84 B1 AD 2F 02 BD 00 D4 20 4C 06A3
83 20 30 B1 4C CA 80 AD 00 06 BD 06 06 AD 01 06 BD 07 06 05B4
AD 02 06 85 CB AD 03 06 85 CC 60 A2 C1 A0 84 20 42 C6 60 087B
A2 05 A0 85 20 42 C6 60 A2 35 A0 85 20 42 C6 60 A2 34 A0 084E
84 20 42 C6 20 46 B1 60 A2 A7 A0 B3 20 42 C6 20 46 B1 60 07CE
A2 ED A0 B3 20 42 C6 20 46 B1 60 A2 7A A0 84 20 42 C6 20 08A9
46 B1 60 AD 09 D2 C9 3F D0 F9 AD 09 D2 C9 15 D0 F9 60 BA 0A99
48 88 F0 0C A2 B2 EA EA EA CA D0 FA A2 00 F0 F1 A2 AE EA 0D2F
EA EA CA D0 FA 68 AA EA 60 08 68 BD 0E 06 78 AD 0E D4 BD 0A69
0D 06 A9 00 BD 0E D4 60 AD 0D 06 BD 0E D4 AD 0E 06 48 28 05EB
60 BD E3 D5 20 2C B3 A9 80 20 84 B3 BD E0 D5 60 AD 11 06 092A
F0 6C 20 11 B3 AD 0B 06 20 74 B3 BD E3 D5 A9 0F 20 84 B3 0806
C0 01 D0 0D A9 03 BD 0F 06 A9 01 BD 10 06 4C E3 B1 A9 0A 069C
BD 0F 06 A9 01 BD 10 06 EE 10 06 CE 10 06 EE 10 06 CE 10 05B9
06 EE 10 06 CE 10 06 EE 10 06 CE 10 06 CE 0F 06 D0 E3 A9 0715
FF BD 0F 06 CE 10 06 D0 D9 EA EA EA 68 60 20 11 B3 AD 09FF
08 06 20 74 B3 BD E3 D5 A9 0F 20 84 B3 20 C7 B1 BD D0 D5 08E3
20 55 B1 BD F0 D5 20 C7 B1 A9 0E 20 84 B3 BD E0 D5 60 AD 09DD
11 06 D0 03 4C 7D B2 A9 90 20 84 B3 20 11 B3 A9 0D 20 84 06A3
B3 20 3C B3 BD F3 D5 BD F7 D5 BD D7 D5 20 C7 B1 AD 00 D3 0B31
BD 09 06 BD F7 D5 BD F3 D5 BD E3 D5 20 2C B3 A9 0C 20 84 09B7
B3 A9 80 20 84 B3 BD E0 D5 60 A9 90 20 84 B3 20 11 B3 A9 0932
0F 20 84 B3 20 C7 B1 A9 0D 20 84 B3 20 3C B3 BD F3 D5 BD 083C
F7 D5 20 C7 B1 AD 00 D3 BD 09 06 BD F3 D5 BD E3 D5 20 2C 0A36
B3 A9 0C 20 84 B3 20 C7 B1 A9 0E 20 84 B3 A9 80 20 84 B3 07F5
BD E0 D5 60 AD 11 06 D0 03 4C 7D B2 A9 90 20 84 B3 20 11 0815
B3 A9 0D 20 84 B3 A9 0F 20 84 B3 20 3C B3 BD F3 D5 BD F7 08F7
D5 BD D7 D5 20 C7 B1 AD 00 D3 BD 09 06 BD F7 D5 BD F3 D5 0840
BD E3 D5 20 2C B3 A9 0E 20 84 B3 A9 0C 20 84 B3 A9 80 20 0817
84 B3 BD E0 D5 60 AD 06 06 BD 00 D3 20 C7 B1 BD F2 D5 BD 0A0B
FA D5 BD F2 D5 AD 07 06 29 3F 20 94 B3 60 A9 38 BD 02 D3 091F
A9 FF BD 00 D3 A9 0C BD 02 D3 60 AD 0B 06 85 D4 AD 0C 06 85 D5 07CA
00 D3 A9 0C BD 02 D3 60 AD 0B 06 85 D4 AD 0C 06 85 D5 20 07CA
AA D9 20 E6 D8 A0 FF CB B1 F3 10 FB 29 7F 91 F3 CB A9 9B 0CAF
91 F3 A6 F3 A4 F4 20 42 C6 60 BD 00 D3 20 C7 B1 BD F3 D5 0B5A
BD FB D5 BD F3 D5 60 BD 00 D3 20 C7 B1 BD F0 D5 BD F8 D5 0CB6
BD F0 D5 60 BD 00 D3 20 C7 B1 BD F1 D5 BD F9 D5 BD F1 D5 0C7B
BD E1 D5 60 20 20 20 4E 61 73 74 61 76 20 6E 61 70 65 74 07AB
69 20 56 63 63 3D 36 56 2C 20 56 70 70 3D 31 32 70 35 56 0549
2E 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 0405
61 20 6B 6C 61 76 65 73 75 21 20 2D 41 2C 42 29 9B 1D 20 059A
20 20 45 50 52 4F 4D 20 6E 65 6C 74 65 20 6E 61 68 72 61 062B
74 20 28 58 3D 32 35 29 2E 20 20 20 20 20 20 20 20 20 20 034F
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 0510
76 65 73 75 21 20 2D 41 2C 42 29 9B 20 20 20 20 53 6E 69 74 05AB
20 6E 61 70 65 74 69 20 56 70 70 3D 35 56 2C 20 56 63 63 0627
3D 35 56 2E 20 20 20 20 20 20 20 20 20 20 20 20 20 20 02D6
28 43 65 68 61 6D 20 6E 61 20 68 6C 61 76 65 73 75 21 20 0654
2D 41 2C 42 29 9B 1D 20 20 20 20 45 50 52 4F 4D 20 6E 61 70 04FF
72 6F 67 72 61 6D 6F 76 61 6E 61 21 20 20 20 20 20 20 20 059E
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 035C
6D 20 6E 61 20 6B 6C 61 76 65 73 75 21 20 2D 41 2C 42 29 05BD
9B 7D A0 A0 D0 D2 CF C7 D2 C1 CD CF D6 C1 CE C9 A0 C5 D0 0E22
D2 CF CD A0 B2 B7 B1 B2 B8 A0 E1 A0 B2 B7 B2 B5 B6 A0 A0 0D79
20 20 20 20 20 20 20 20 20 20 A0 A0 A0 C8 F5 ED E2 CE E5 A0 D3 09B0
EF E6 F4 F7 E1 F2 E5 A0 A0 1D 9B 20 50 50 7C 6F 67 72 0B5A
61 6D 6F 76 61 6E 69 20 45 50 52 4F 4D 20 32 37 31 32 38 05B2
20 28 31 36 6B 42 29 21 20 20 20 20 1D 1D 50 6F 6B 79 6E 0471
79 3A 9B 20 20 50 72 6F 67 72 61 6D 6F 76 61 6E 69 20 45 06EB
50 52 4F 4D 20 32 37 32 35 36 20 28 33 32 6B 42 29 21 20 0428
20 20 20 1D 1D 50 6F 6B 79 6E 79 3A 9B 03F9
```

```

10 REM *****
20 REM *   OBSLUZNY PROGRAM PRO   *
30 REM *   NAHRAVANI EPROM       *
40 REM * 11.7.88. Humble Software *
50 REM *****
60 ? "Programovani 16/32 kB EPROM (zadej 16 nebo 32) ... ";
70 INPUT DRUH
80 IF DRUH<>16 AND DRUH<>32 THEN 60
90 DRUH=DRUH/16-1
100 ? "Adresa EPROM, od které se zacina prog-ramovat ...";
110 INPUT ZACEP
120 ? "Adresa RAM, kde zacinaji data pro prog-ramovani EPROM ..";
130 INPUT ZACRAM
140 ? "Adresa RAM, kde konci data pro progra-movani ...";
150 INPUT KONRAM
160 KONRAM=KONRAM+1
170 POKE 1536, ZACEP-INT(ZACEP/256)*256
180 POKE 1537, INT(ZACEP/256)
190 POKE 1538, ZACRAM-INT(ZACRAM/256)*256
200 POKE 1539, INT(ZACRAM/256)
210 POKE 1540, KONRAM-INT(KONRAM/256)*256
220 POKE 1541, INT(KONRAM/256)
230 POKE 1553, DRUH
240 ? "Kontroluj napeti Vcc a Vpp (+5V) na pameti EPROM!
      (Cekam na klavesu!)"

250 POKE 764, 255
260 IF PEEK(764)=255 THEN 260
270 C=USR(32768)
280 ? " KONEC!"
290 END
30000 POKE 54018, 56:POKE 54016, 255:POKE 54018, 60
30010 POKE 54016, 128
30020 POKE 54768, 0
30030 POKE 54776, 0
30040 POKE 54768, 0
30050 POKE 54016, 0
30060 POKE 54771, 0
30070 POKE 54779, 0
30080 POKE 54771, 0
30090 POKE 54770, 0
30100 POKE 54778, 0
30110 POKE 54770, 0
30115 POKE 54016, 128+64
30120 POKE 54769, 0
30130 POKE 54777, 0
30140 POKE 54769, 0

```

Výpis 1. Základní program v jazyku BASIC (915-V1)

ce. Po úspěšném naprogramování paměti EPROM se na displeji zobrazí nápis „Sniz napeti Vpp=5 V, Vcc=5 V. (Cekam na klavesu! – A, B)“.

Po snížení programovacích napětí (v uvedeném pořadí) na určené hodnoty a stisknutí kláves A a B program provádí verifikaci naprogramovaných bajtů. Pokud nebyla nalezena chyba, zobrazí se na displeji nápis „EPROM naprogramovana. (Cekam na klavesu! – A, B)“.

Po stisknutí uvedených kláves následuje návrat do programu v BASIC a nápis „KONEC!“.

Tímto postupem je celá požadovaná oblast paměti EPROM naprogramována.

V případě, že ani po 25 jednomilisekundových impulsích není do paměti EPROM správně zapsán požadovaný bajt, objeví se nápis

„EPROM nelze nahrát (X=25). (Cekam na klavesu! – A, B)“.

Po stisku kláves A, B se vrátíte do programu v BASIC a zobrazí se nápis

„KONEC!“.

V případě, že při verifikaci po skončení programování při provozních hodnotách napětí byl v paměti EPROM nalezen bajt, který neodpovídá příslušnému bajtu v RAM počítače, na obrazovce se objeví stejný text, jako při chybném zápisu po 25 impulsích délky 1 ms a po stisku kláves A, B se program vrátí do BASIC a zobrazí se nápis

„KONEC!“.

Před zasunutím a vyjmutím EPROM z obímky přípravku se nastaví logická 0 na přívodech EPROM příkazem

GOTO 30000 + RETURN.

## Závěr

Zvolené hardwarové uspořádání umožňuje programovat EPROM 27128 (16 kB) nebo 27256 (32 kB). V případě EPROM 27256 se nejdříve programuje 16 kB a po změně úrovně na adrese A14 dalších 16 kB.

Programování paměti EPROM jsme ověřili s paralelním interfejsem využívajícím joystickové porty a port pro programový modul (popsaný v [1]). Zápis 16 kB včetně verifikace trvá asi 4 min.

## Literatura

- [1] Pokorný J. et al.: Univerzální paralelní interfejs k počítačům ATARI 800XL a 130XE. Zasláno do Amatérského Radia.

# PROGRAMÁTOR EPROM 8708

## ŘÍZENÝ ŠKOLNÍM MIKROPOČÍTAČEM PMI-80

Ing. Jaroslav Pipek, Horská 439, 543 02 Vrchlabí

Článek se zabývá popisem programovacího přípravku přímo slučitelného se školním mikropočítačem PMI 80. Obvodové řešení však může být vodítkem i pro vlastníky jiných mikropočítačů, kteří nemají možnost paměti EPROM programovat. Přestože PMI 80 patří mezi nejjednodušší, lze jej vedle praktického procvičování instrukčního souboru mikropočítače MHB8080 úspěšně využít jako základní jednotku pro realizaci celé řady i poměrně složitých automatizačních a regulačních zařízení. Je ovšem nutné tuto základní jednotku patřičně rozšířit jak obvodově, tak programově v souladu s danou aplikací.

Při návrhu a oživování zařízení se neobejdeme (alespoň v počátečním stadiu) bez použití paměti typu EPROM, které umožňují trvale uchovat informaci bez ohledu na jejich zapojení do obvodu. Lze je však vymazat ozářením ultrafialovými paprsky. To umožňuje použití těchto pamětí i všude tam, kde již program je funkční, ale kde se dá předpokládat, že časem dojde k jeho obměně.

Navržený přípravek umožní uživateli PMI 80 pohotově kdykoli změnit obsah těchto pamětí nezávisle na jiném systému.

## Popis přípravku

Přípravek je osazen pouze dvěma IO (MH7400 a UCY74123) a šesti tranzistory.

Odpadá adresový i datový registr. Jako registr je využito údajů na datové a adresní sběrnici ve stavu WAIT mikropočítače. Celý přípravek je sestaven na oboustranně plátované kupřetřítové destičce o rozměrech 81 × 80 mm a zasouvá se do konektoru K1 na desce PMI-80.

## Obsluha a činnost

Nejprve jsou data určená k zápisu do EPROM zapsána z klávesnice PMI-80 do zóny RAM paměti (zvolena oblast 1C00 až 1CFF). Tento blok dat lze přepsat ze zóny RAM do libovolné oblasti programované paměti EPROM v rozsahu 0000 až 03FF. Počáteční adresu této oblasti předem uložíme z klávesnice do pomocných buněk 1F00 a 1F01 v oblasti paměti RAM PMI-80. Přitom do buňky 1F00 ukládáme nižší bajt této adresy v hexadecimálním tvaru, vyšší bajt do buňky 1F01. Údaje vyššího bajtu však korigujeme vždy o + 20H (viz tab.1) s ohledem na aktivaci programované EPROM signálem A13. Do další pomocné buňky 1F02 ukládáme v hexadecimálním tvaru údaj o délce bloku dat v RAM (jinak řečeno obsah 1F02 udává nižší bajt koncové adresy bloku dat v RAM). S ohledem na skutečnost, že jsme pro data určená k přepsu do EPROM vymezili v RAM oblast 1C00 až 1CFF, je délka bloku dat max. 256 bajtů. Jednoduše lze však naprogramovat až 1 kB dat, bude-li se opakovat programovací cyklus (s novou

náplní buněk 1F00 až 1F02 podle **tab.1)** a s novým blokem dat umístěným opět do zóny 1C00 až 1CFF celkem čtyřikrát. Je to nutné vzhledem k nedostupnosti dalšího 1 kB paměti RAM. Bude-li k dispozici celý 1 kB RAM pouze pro účel programování, pak lze programovat EPROM najednou (ovšem s patřičně změněným řídicím programem). Řídicí program je umístěn v pomocné paměti EPROM, která je zasunuta do volné pozice na desce PMI-80 (její počáteční adresa je 0400). Programovaná paměť se umísťuje na desku programátoru. Programování (přepis z RAM) začíná spuštěním řídicího programu „PROGRAM“ na adrese 0400 (řídicí program lze umístit libovolně, podle toho se mění startovací adresa). Data určená pro zápis do EPROM jsou postupně načítána ze zóny paměti RAM (1C00 až 1CFF) a přesunovou instrukcí MOV M, A prostřednictvím bitu A13 adresní sběrnice je startován monostabilní klopný obvod MKO1, který vytváří signál READY a uvede mikroprocesor do stavu WAIT. Současně je spínacím obvodem, tvořeným tranzistorem T5 a T6, přivedena na vstup CSn/WE úroveň 12 V. Stav adresní sběrnice se nemění, data na datové sběrnici jsou připravena od okamžiku přechodu signálu WR na log.0. Současně s přechodem signálu WAIT na úroveň H je přes zpozdovací obvod, tvořený dvěma hradly MH7400, spuštěn druhý monostabilní klopný obvod MKO2, který generuje impuls pro připojení programovacího napětí  $U_{prog}$  spínačem s tranzistorem T1 a T2. Je-li programování ukončeno, na displeji se rozsvítí nápis END. Tlačítkem RESET přejdeme do výchozího stavu mikroprocesoru, doprovázeného nápisem PMI-80.

Správnost zapsaných dat mohou okamžitě přezkontrolovat spuštěním programu „KOMPARACE“ na adrese 0430. Je-li některý bajt zapsán chybně, zobrazí se v datovém poli displeje číslo tohoto bajtu. Byl-li celý blok naprogramován správně, rozsvítí se nápis „C dobře“. Obsah programované paměti lze též kdykoli přecíst běžným způsobem počínaje adresou 2C00. K tomuto účelu slouží spínač z tranzistorů T3 a T4.

## Další možnosti řídicího programu

Často potřebujeme přeprogramovat obsah jedné paměti EPROM do druhé. V tomto případě si pomůžeme následovně, aniž bychom museli vyvádět sběrnice. První paměť (ORIGINAL) umístíme do objímky, připájené shora k vývodům na paměti MHB8608 s programem MONITOR. Pouze vývod CSn je propojen lankem s pozicí CS4n na konektoru K1. Druhá EPROM (KOPIE), na niž bude pořizována kopie, je v objímce na desce programátoru.

Spuštěním uživatelského programu KOPIE na adrese 0410 se přepíše celá paměť z pozice ORIGINAL na pozici KOPIE (program testuje stavy FF, které přeskakuje). V tomto případě může být obsah pomocných buněk 1F00 až 1F02 libovolný. Program tyto buňky nerespektuje.

V případě, že potřebujeme změnit obsah dat v EPROM, umístíme tuto do pozice ORIGINAL. Na pozici kopie umístíme paměť prázdnou. Spuštěním programu PREPIS na adrese 0470 (v souladu s **tab. 1**) se přepíše požadovaný blok dat do oblasti RAM na adresy 1C00 až 1CFF, kde je možné běžným způsobem (viz obsluha PMI 80) změnit obsah požadovaných buněk. Blok dat po opravě přepíšeme do EPROM spuštěním již známého programu PROGRAM na adrese 0400. Tuto možnost přechodu přes RAM lze využít i pro přemístění bloku dat z dané pozice původní EPROM do libovolné oblasti nové EPROM.

Tab. 1. Pomocné buňky v paměti RAM (914–T1)

1F00 obsahuje nižší bajt adresy, udávající počátek bloku dat v EPROM, a to:	
a) v programované EPROM při programu "PROGRAM",	
b) ve snimané EPROM při programu "PŘEPIS".	
1F01 obsahuje vyšší bajt, závisí na oblasti EPROM, kterou programují při programu PROGRAM, nebo z které snímán při programu PŘEPIS.	
programovaná oblast přepisovaná oblast EPROM	obsah 1F01
0000 - 00FF	20
0100 - 01FF	21
0200 - 02FF	22
0300 - 03FF	23

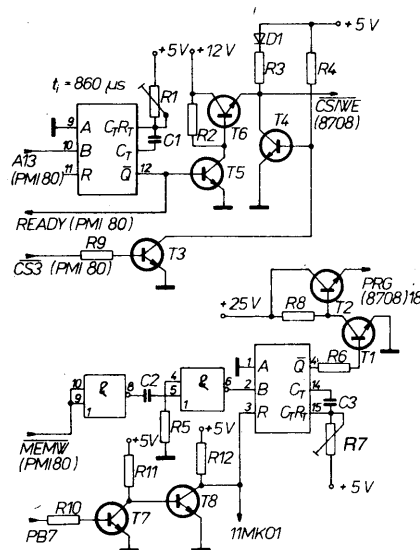
### Poznámka

Jelikož není použit adresový dekodér, ale přímo bit A13 na přechod do stavu WAIT, mohlo by v důsledku činnosti programu MONITOR dojít k přepsání již naprogramovaných správných dat při ponechání programovacího napětí + 25 V. Proto je využit výstup PB7 IO10 pro blokování obou MKO mimo vlastní program „PROGRAM“. Oba jsou blokovány vždy od okamžiku připojení zdroje k PMI-80 a po stisku tlačítka RESET (neboť bit PB7 brány PB IO10 je ve třetím stavu, tudíž T8 je sepnut). Programově je zajištěno, že pouze při programování přechází výstup PB7 do stavu H a tím uvolňuje činnost obou MKO (UCY74123). Vývod PB7 je z pozice 41 konektoru K2 spojen lankem na vstup T7. Signál z kolektoru T8 je přiveden na nevyužitou pozici 28 konektoru K1. Obvod s tranzistorem T7 a T8 je umístěn v univerzálním poli přímo na desce PMI-80.

## Oživení a nastavení

Zapojení je natolik jednoduché, že s oživením přípravku nebudou problémy. Po osazení a přezkontrolování vodivých cest podle **obr. 1** nastavíme běžce trimrů asi do poloviny dráhy.

První známka správné funkce bude, když po zasunutí přípravku do konektoru K1 (při přerušeném vývodu z kolektoru T8 na nulovací vstupy UCY74123) bude blikat nápis



Obr. 1. Schéma přípravku (914-1)

„PMI-80“. Podle **1** by měla být délka programovacího pulsu  $t_{pw}$  v rozmezí 0,1 až 1 ms. Pro spolehlivé programování paměti musí být současně splněna nerovnost  $N \cdot t_{pw} \geq 100$  ms, kde N je počet programovacích cyklů.

Volíme  $N = 128$  (viz příkaz **MVI B,80** na adrese 00FB),  $t = 800 \mu s$ . Pomocí osciloskopu nastavíme trimrem R7 dobu kyvu MKO 2 asi na 800  $\mu s$ , trimrem R1 dobu kyvu MKO 1 asi na 860  $\mu s$ . Zapojení s těmito časy funguje spolehlivě. Dále připojíme programovací napětí. Na výstupu emitoru T2 se musí objevit puls s úrovní + 25 V a délkou 800  $\mu s$ . Po připojení vývodu z kolektoru T8 na nulovací vstupy UCY74123 je přípravek připraven k činnosti. Zbývá ještě uložit do pomocné paměti EPROM vlastní řídicí program podle **Výpisu 2**. Jedna z možností, jak tuto paměť již v tomto stadiu naprogramovat bez použití jiného programátoru, je umístit řídicí program „PROGRAM“ (zahrnuje části začínající adresami: PROGRAM, PRESUN, ZOBRAZ) např. počínaje adresou 1D00 do paměti RAM. Nyní můžeme již známým způsobem uložit celý řídicí program podle **Výpisu 2** s tím, že startovací adresa je 1D00 (pozor též na změnu adres ve skokových instrukcích).

Vlastní programování si názorně objasníme na dvou příkladech. Představme si nejprve, že je třeba do paměti EPROM počínaje adresou např. 01AA zapsat následující blok dat: F3, 3E, 20, 06, 80, 21, FF, 1E, 23, 00, 76. Postupně provedeme následující kroky:

- 1) Data zapíšeme do paměti RAM počínaje adresou 1C00:

1C00 F3  
1C01 3E  
1C02 20  
1C03 06  
1C04 80  
1C05 21  
1C06 FF  
1C07 1E  
1C08 23  
1C09 00  
1C0A 76

Délka bloku dat, jak je zřejmé, je 0A.

- 2) V souladu s **tab. 1** zapíšeme na pomocné buňky:

1F00 AA (nižší bajt počáteční ukládací adresy)  
1F01 21 (vyšší bajt + 20H)  
1F02 0A (délka bloku dat)

- 3) Odstartujeme program PROGRAM příkazem G.

**Výpis 1.** Program v symbolických instrukcích 8080. Startovací adresa = absolutní adresa EPROM + 0400H (paměť je aktivována vstupem CS1n) (914-V1)

(914-V1)

Adresa  
absolutní

```

0000 PROGRAM: MVI C,FF ; pomocná data
0002 LHL D,1F00 ; vytvoření počáteční ukládací adresy
0005 LXI D,1C00 ; počátek bloku dat v RAM
0008 JMP PRESUN

0010 KOPIE: LXI D,1000 ; počátek bloku dat v EPROM
          "ORIGINAL"
0013 LXI H,2000 ; počáteční adresa v EPROM
          "KOPIE"
0016 MVI C,00 ; pomocná data
0018 JMP PRESUN

0020 PRESKOC 1: LDA 1F02 ; celý blok naprogramován?
0023 CMP L
0024 JNZ TU ; není
0027 JMP KONEC ; ano

0030 KOMPARACE: CALL VSTUP
0033 ORI OC ; vytvoření počáteční adresy EPROM
0035 MOV H,A,
0036 LXI D,1C00 ; poč.adresa RAM
0039 XCHG
003A DCX H
003B DCX D
003C OPET: INX H
003D INX D
003E DCX B ; test délky bloku dat
003F MOV A,C
0040 CMP B
0041 JZ ANO ; data zapsána správně
0044 LDAX D
0045 CMP M ; není chyba?
0046 JNZ OPET ; testuj další bajt
0049 MOV A,L ; číslo chybného bajtu
004A STA 1FFA
004D CALL 00FB ; zobraz číslo chybného bajtu

0055 VSTUP: LDA 1F02
0058 MOV C,A ; nastavení čítače cyklu
0059 MVI B,00
005B INX B
005C INX B
005D LHL D,1F00
0060 MOV A,H ; utvoř vyšší bajt počáteční adresy, přepiš z pozice 1000 do RAM

0070 PREPIS 1: CALL VSTUP
0073 ORI 10 ; realizace adresy "ORIGINAL"
0075 JMP VYSTUP

0080 PREPIS 2: CALL VSTUP ; realizace adresy program!EPROM
0083 ORI OC,
0085 JMP VYSTUP

0090 VYSTUP: MOV H,A
0091 LXI D,1C00
0094 DCX H
0095 DCX D
0096 TAM: INX H
0097 INX D

```

```

0098 DCX B
0099 MOV A,C
009A CMP B
009B JZ ZOBRAZ ; celý blok dat je přepsán?
009E MOV A,M ; ne
009F XCHG
00A0 MOV M,A
00A1 XCHG
00A2 JMP TAM ; pokračuj v prepisu dat

00AA ANO: LXI H,190C ; zobraz "C dobre"
00AD SHLD 1FFF ; ukládej kódy znaku do výst. registru PMI 80

00B0 LXI H,110D
00B3 SHLD 1FF1
00B6 LXI H,120B
00B9 SHLD 1FF3
00BC LXI H,190E
00BF SHLD 1FF5
00C2 MVI A,19
00C4 STA 1FF7
00C7 CALL 0116 ; podprogram "OUTKE" PMI 80

00D0 ZOBRAZ: LXI H,1919 ; zobraz "END"
00D3 SHLD 1FFF ; ukládej kódy znaku do výst. registru PMI 80

00D6 LXI H,0E19
00D9 SHLD 1FF1
00DC LXI H,0D1B
00DF SHLD 1FF3
00E2 LXI H,1919
00E5 SHLD 1FF5
00E8 MVI A,19
00EA STA 1FF7
00ED CALL 0116 ; podprogram "OUTKE" PMI 80

00F0 PRESUN: MVI A,88 ; programování 82 55
00F2 OUT FB
00F4 MVI A,80 ; PB7 na úroveň "H"
00F6 OUT F9
00F8 XCHG
00F9 DCX H
00FA DCX D
00FB TU: MVI B,80
00FD INX H ; přechod na další adresu
00FE INX D
00FF MOV A,M ; přesun dat z RAM
0100 CPI FF
0102 JZ PRESKOC ; neprogramuj "FF"
0105 SEM: XCHG
0106 MOV M,A ; cyklus programování jednoho bajtu dat

0107 DCR B
0108 JNZ SEM ; je cyklus ukončen?
010B XCHG,ANO
010C PRESKOC: MOV A,C ; je požadován program nebo kopie?

010D CPI FF
010F JZ PRESKOC 1; program
0112 NOP
0113 NOP
0114 MOV A,L ; kopie
0115 CPI FF ; test ukončení prepisu
0117 JNZ TU
011A MOV A,H
011B CPI 13
011D JNZ TU ; celá paměť překopírována?
0120 KONEC: MVI A,00 ; ano
0122 OUT F9
0124 JMP ZOBRAZ ; zobraz nápis "END"

```

Nyní si povšímneme složitějšího případu, když přechází blok dat svojí délkou přes dvě oblasti. V tom případě musíme program rozdělit na dva dílčí úseky. Např. blok dat z prvního příkladu chceme umístit do EPROM počínaje adresou 00FB. Podle **tabulky 1** je zřejmé, že se přechází z oblasti 1 do oblasti 2. Provedeme posloupnost následujících kroků:

- 1) Určíme délku bloku dat, který lze ještě umístit do zóny 1 pomocí vztahu:  
**FF – nižší bajt počáteční ukládací adresy + 01.**  
Tedy  $FF - FB + 01 = 05$ .

- 2) Buňky v RAM obsadíme následovně:  
**1C00 F3**  
**1C01 3E**  
**1C02 20**  
**1C03 06**  
**1C04 80**

- 3) V souladu s tab. 1 запиšeme na pomocné buňky:  
**1F00 FB** (nižší bajt počáteční ukládací adresy)  
**1F01 20**  
**1F02 04** (délka bloku dat)

- 4) Odstartujeme program PROGRAM.

- 5) Zbýlá data запиšeme do RAM opět počínaje buňkou 1C00:  
**1C00 21**  
**1C01 FF**

## Výpis 2. Obsah pomocné paměti EPROM s řídicím programem (914–V2)

### Seznam součástek

R1,R7	TP 011	22 kΩ
R2,R4	TR 151	12 kΩ
R3,R5	TR 151	390 Ω
R6	TR 151	5,6 kΩ
R8	TR 151	6,8 kΩ
R9–R12	TR 151	4,7 kΩ

C1,C3	TK 782	150 nF
C2	TK 782	10 nF

T1, T2, T4, T6,	KC507
T3, T7, T8	KC508
T5	KSY63

**1C02 1E**  
**1C03 23**  
**1C04 00**  
**1C05 76**

- 6) Počáteční ukládací adresa je nyní 0100 a v souladu s **tab. 1** uložíme na:

**1F00 00**  
**1F01 21**  
**1F02 05**

- 7) Odstartujeme program PROGRAM příkazem G.

0000	0E FF 2A 00 1F 11 00 1C C3 F0 04 FF FF FF FF FF
0010	11 00 10 21 00 20 0E 00 C3 F0 04 FF FF FF FF FF
0020	3A 02 1F BD C2 FB 04 C3 20 05 FF FF FF FF FF
0030	CD 55 04 F6 0C 67 11 00 1C EB 2B 1B 23 13 0B 79
0040	B8 CA AA 04 1A BE CA 3C 04 7D 32 FA 1F CD FB 00
0050	FF FF FF FF FF FF 3A 02 1F 4F 06 00 03 03 2A 00 1F
0060	7C C9 FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0070	CD 55 04 F6 10 C3 90 04 FF FF FF FF FF FF FF FF
0080	CD 55 04 F6 0C C3 90 04 FF FF FF FF FF FF FF FF
0090	67 11 00 1C 2B 1B 23 13 0B 79 B8 CA D0 04 7E EB
00A0	77 EB C3 96 04 FF FF FF FF FF FF 21 0C 19 22 EF 1F
00B0	21 0D 11 22 F1 1F 21 0B 12 22 F3 1F 21 0E 19 22
00C0	F5 1F 3E 19 32 F7 1F CD 16 01 FF FF FF FF FF FF
00D0	21 19 19 22 EF 1F 21 19 0E 22 F1 1F 21 1B 0D 22
00E0	F3 1F 21 19 19 22 F5 1F 3E 19 32 F7 1F CD 16 01
00F0	3E 88 D3 FB 3E 80 D3 F9 EB 2B 1B 06 80 23 13 7E
0100	FE FF CA 0C 05 EB 77 05 C2 06 05 EB 79 FE FF CA
0110	20 04 00 00 7D FE FF C2 FB 04 7C FE 13 C2 FB 04
0120	3E 00 D3 F9 C3 D0 04 FF FF FF FF FF FF FF FF FF

Při přepisu z paměti EPROM v pozici ORIGINAL do paměti RAM je postup úplně shodný s tím, že na 1F00 ukládáme opět nižší bajt počáteční adresy, tentokrát však přepisovaného bloku dat, na 1F01 vyšší bajt této adresy zvětšený opět o + 20H a na 1F02 opět délku přepisovaného bloku dat.

### Literatura

- 1) Programátor paměti EPROM řízený mikropočítačem. ST8/81.

# OSCILOSKOP ZO ZX SPECTRA

Jindřich Videnský, Lamač-Podháj 55, 841 03 Bratislava

Program bol vytvorený pre využitie AD prevodníka s mikropočítačom v elektrotechnickej praxi ako jednoduchého nf pamäťového osciloskopu. Umožňuje merať v ôsmich pamäťových rozsahoch signály do 15 kHz.

Po nahrani programu do počítača sa zobrazí základné menu:

- F...** Zmena farieb INK, PAPER, BORDER.  
**Z...** Zmena riadiaceho slova, jeho adresy a adresy vstupného portu.  
**P...** Zmena adresy vlastného podprogramu, ľubovoľnej rutiny, ktorá sa dá vyvolať počas behu osciloskopu. Môže to byť napr. podprogram pre Hardcopy.  
**C...** Zmena synchronizačnej hodnoty (viď ďalej).  
**S...** Štart osciloskopu. Stlačením tejto klávesy sa vypíše

hlavné menu s týmito možnosťami

- 1–8** Stlačením jednej z týchto kláves vyberieme časový rozsah a zároveň sa spustí zobrazovací režim osciloskopu, v ktorom prebieha vzorkovanie signálu a jeho zobrazovanie.  
**Z...** V zobrazovacom režime sa vykreslí na obrazovku sieťka 5×10.

- X...** Sieťka sa nebude vykresľovať.  
**C...** Program po vykreslení priebehu čaká na stlačenie ľubovoľnej klávesy, čím sa obraz na obrazovke zastaví a je možné napr. presne odčítať amplitúdu, alebo urobiť kópiu obrazovky.  
**V...** Program nečaká na stlačenie klávesy, takže obraz sa mení plynule, podobne ako na obyčajnom osciloskope.  
**B...** Vzorkovanie bude prebiehať bez softwarovej synchronizácie, viď N.  
**N...** Synchronizácia priebehu vzorkovania prechodom signálu cez hodnotu nastavenú v základnom menu klávesou C. Program čaká, kým signál nedosiahne nastavenú hodnotu, zistí či priebeh rastie a ak áno, potom ho zobrazí. Tým sa dosiahne, že signál nebehá po obrazovke, ale vždy začína v určitej hodnote. Využiť sa dá tento režim hlavne pri periodických signáloch.  
**M...** Tak isto, ako pri N, ale synchronizácia klesajúcim priebehom.  
**W...** Návrat do základného menu.

- P...** Stlačením tejto klávesy počas zobrazovacieho režimu sa vykoná odskok na vlastný podprogram, ukončený inštrukciou RET. Zmenu adresy v základnom menu je vhodné urobiť len vtedy, ak je podprogram skutočne v pamäti, ináč hrozí zrušenie systému pri náhodnom stlačení P. Podprogram je treba umiestiť nad adresu 41500.  
**Q...** Po stlačení Q sa dá vrátiť do tohoto menu zo zobrazovacieho režimu.

Program pracuje na princípe navzorkovania signálu do pamäti tak, aby dĺžka vzorky zodpovedala danému časovému rozsahu. Preto každému rozsahu zodpovedá jeden z časovacích podprogramov F01 až F680. Tieto zaisťujú pravidelné intervaly medzi jednotlivými vzorkami signálu. Touto časťou je limitovaná aj najvyššia frekvencia, ktorú je možné zobraziť, rešp. ešte odčítať na obrazovke. Na obrázku je príklad signálu 15 kHz, ktorý je ešte rozlíšiteľný. Po navzorkovaní 256 hodnôt z AD prevodníka do pamäti sú vzorky prepočítané z intervalu 9 až 255 do intervalu 0 až 191, aby ich bolo možné zobraziť na obrazovku Spectra.

Program sa skladá z BASICu a strojového kódu. Neodporúčam robiť v ňom veľké zmeny, hlavne riadok 700 musí zostať presne tak ak je, tzn. vcelku, dodržať všetky čiarky a pred riadkom 700 sa nesmie vyskytnúť iný riadok s príkazom DATA.

Strojový kód je písaný v assembleri, je odladený systémom MRS. Odporúčam nerobiť v ňom žiadne zmeny, preložiť ho od adresy 40000, pretože je modifikovaný príkazmi POKE z BASICu. Takisto je treba v assembleri preložiť aj inštrukcie NOP, ktoré sa na prvý pohľad môžu zdať zbytočné. Pretože je výpis programu vytlačený malými písmenami, je treba dáť pozor pri prepisovaní na zámenu písmena l s číslom 1.

## Technické predpoklady

Je treba použiť osembitový A/D prevodník, ktorý má automatický štart prevodu a vyrovnávaciu pamäť. Vhodný je napríklad prevodník zo zelenej prílohy AR Mikroelektronika 1988. Rychlosť prevodu by však mala byť menšia ako  $5,7 \mu s$ . Je daná najrýchlejším rozsahom a najrýchlejšou inštrukciou blokového prenosu s opakovaním INIR. Pri použití prevodníka s menším počtom bitov ako 8 sa zbytočne znižuje rozlišovacia schopnosť osciloskopu a použitie napr. 12 bitového prevodníku s využitím jeho horných 8 bitov je síce možné, avšak je zbytočným luxusom vzhľadom k rozlišovacím schopnostiam obrazovky Spectra. Citlivosť osciloskopu je daná citlivosťou prevodníku.

Najvhodnejšie sú prevodníky typu MDAC 08, program nie je určený pre prevodníky s multiplexovým BCD výstupom typu C520. Vhodné je, aby vstup prevodníku bol bipolárny, vzhľadom k potrebe sledovať nízke signály.

Na priložených kópiách obrazovky sú nasnímané rôzne priebehy v rôznych časových rozsahoch. Je z nich vidieť príklad použitia synchronizácie priebehu prechodom cez "nulovú" hodnotu, použitie sieťky, ako aj kvalita generátora „sinusového“ priebehu.

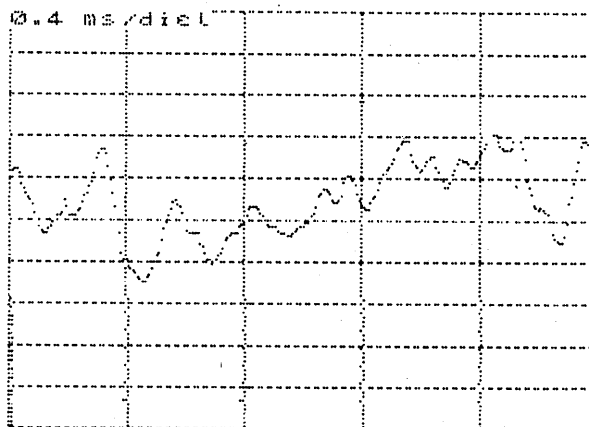
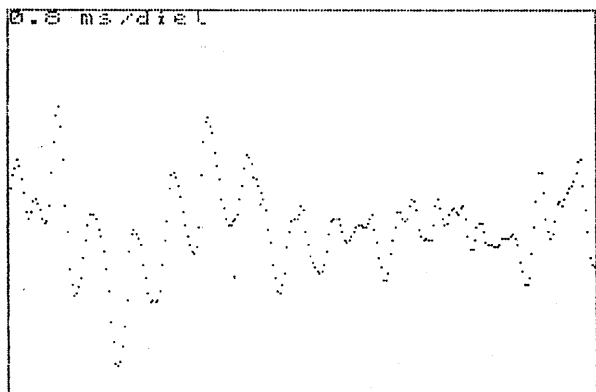
## Záver

Program osciloskop nebol určený ako náhrada skutočného osciloskopu, ktorému sa svojimi parametrami nemôže rovnať, ale iba ako pomerne užitočná pomôcka pri nastavovaní jednoduchých elektrotechnických zariadení, na rôzne orientačné sledovanie analógových aj digitálnych signálov a ako náhrada za logickú sondu.

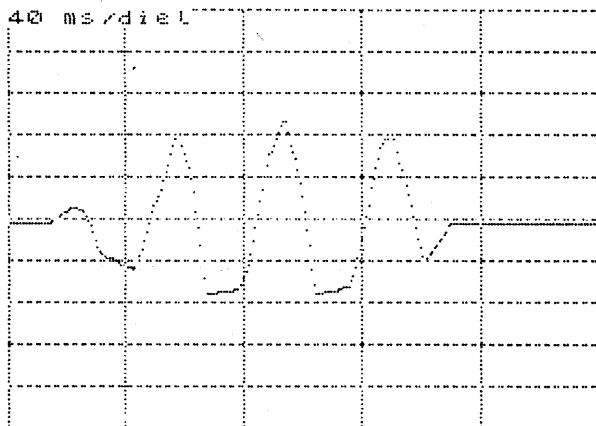
## Výpis 1. Program v BASICu (928-V1)

```
10 CLEAR 39999: BORDER 1: PAPE
R 1: INK 7: CLS : LOAD ""CODE 40
000: CLS : BORDER 0
20 LET ash=40472: LET sh=127:
LET ash2=40478: LET ars=127: LET
rs=130: LET vp=63: LET op=31
40 GO TO 600
50 PRINT AT 3,0;"1-ZMENA ADRES
Y RIADIACEHO SLOVA PRE INTERFA
CE"
60 PRINT AT 5,0;"2-ZMENA RIADI
ACEHO SLOVA PRE IN-TERFACE"
70 PRINT AT 8,9;"OUT(";ars;"),
";rs;"
80 PRINT AT 10,0;"3-ZMENA ADRE
SY VSTUPNEHO PORTU PREVODNIKU
(A/D PREVOD)"
90 PRINT AT 13,13;"IN(";PEEK 4
0707;")"
100 PRINT AT 15,0;"4-ZMENA ADRE
SY VYSTUPNEHO PORTU PREVODNIKU
(D/A PREVOD NIE JE NUTNY)"
110 PRINT AT 18,13;"OUT(";op;")"
120 PRINT AT 21,0; BRIGHT 1;"R-
NAVRAT K MENU"
130 IF INKEY$="1" THEN INPUT a
rs: GO TO 50
140 IF INKEY$="2" THEN INPUT r
s: GO TO 50
150 IF INKEY$="3" THEN INPUT v
p: POKE 40707,vp: GO TO 50
160 IF INKEY$="4" THEN INPUT o
p: GO TO 50
170 IF INKEY$="R" OR INKEY$="r"
THEN GO TO 460
180 GO TO 130
400 INPUT "INK ";A: IF A<0 OR A
>7 THEN GO TO 400
410 INK A
420 INPUT "PAPER ";A: IF A<0 OR
A>7 THEN GO TO 420
430 PAPER A
```

```
440 INPUT "BORDER ";A: IF A<0 O
R A>7 THEN GO TO 440
450 BORDER A
460 CLS
600 PRINT AT 6,1;"F-ZMENA FARIE
B";;"Z-ZMENA HODNOTY PRE INTERFA
CE";;"P-ZMENA ADRESY VLASTNEHO
";;"PODPROGRAMU";;"C-ZMENA SYN
CHRONIZACNEJ HODNOTY"; BRIGHT 1
;"S-START OSCIOSKOPU"
610 IF INKEY$="F" OR INKEY$="f"
THEN GO TO 400
620 IF INKEY$="Z" OR INKEY$="z"
THEN CLS : GO TO 50
630 IF INKEY$="S" OR INKEY$="s"
THEN GO TO 700
640 IF INKEY$="C" OR INKEY$="c"
THEN INPUT "nova hodnota ";sh:
POKE ash,sh: POKE ash2,sh
650 IF INKEY$="P" OR INKEY$="p"
THEN INPUT "adresa nesmie byt
z intervalu 40000-41500 ";ppp:
POKE 40372,INT (ppp/256): POKE
40371,ppp-256*INT (ppp/256): GO
TO 610
660 GO TO 610
700 DATA "1- 2 s/diel,2- 200
ms/diel,3- 40 ms/diel,4- 8 m
s/diel,5- 2 ms/diel,6- 0.8 ms/
diel,7- 0.4 ms/diel,8- 0.3 ms/di
el,Z-so sieťkou,X-bez sieťky,C-c
äka na stlačenie klavesy, po vy
kreslení,V-necaka na stlačenie k
lavesy,B-vzorkovanie bez synchro
nizacie,N-synchronizacia prechod
om cez, nastavenu hodnotu stupa
jucim, priebehom,M-synchronizac
ia klesajucim, priebehom,W-navr
at do hlavneho menu,P-vystup na
vlastny podprogram,Q-navrat zo z
obrazovacieho, režimu do tohoto
menu, ♦HEAVYSOFT 1988"
710 OUT ars,rs: RANDOMIZE USR 4
0000
720 GO TO 460
1000 SAVE "OSCIOSKOP" LINE 10
```



Obr. 1. Ukážky vytisknutých průběhů (928-1)



## Výpis 2. Rutina ve strojovém kódu (928-V2)

3	60bb	*a	org 40000
74	60bb		
5	9c40		
6	9c40		
7	9c40		
8	9c40 f3		
9	9c41 cd8a9d		
10	9c44 cda09d		
11	9c47 cd419d		
12	9c4a e5		
13	9c4b dde1		

;vstupna rutina,tlac menu  
 ;modifikacia programu  
 osc di  
 call clear  
 call vstup  
 call data  
 push hl  
 pop ix



14	9c4d cd559d	scan1	call prnt	110	9d0f d5	push de
15	9c50 cd059e	scan	call keys	111	9d10 cdf19d	call prep
16	9c53 cd1b2d		call #2d1b	112	9d13 d1	pop de
17	9c56 304b		jr nz,vzor	113	9d14 4a	ld c,d
18	9c58 fe5a		cp 'Z'	114	9d15 cdb022	call #22b0
19	9c5a 2004		jr nz,xx	115	9d18 3c	inc a
20	9c5c af		xor a	116	9d19 47	ld b,a
21	9c5d 32b69d		ld (gridd),a	117	9d1a 3e01	ld a,1
22	9c60 fe58	xx	cp 'X'	118	9d1c 0f	lop rrca
23	9c62 2005		jr nz,cc	119	9d1d 10fd	djnz lop
24	9c64 3ec9		ld a,#c9	120	9d1f ae	xor (hl)
25	9c66 32b69d		ld (gridd),a	121	9d20 77	ld (hl),a
26	9c69 fe43	cc	cp 'C'	122	9d21 e1	pop hl
27	9c6b 2004		jr nz,vv	123	9d22 14	inc d
28	9c6d af		xor a	124	9d23 c8	ret z
29	9c6e 32049e		ld (keyp),a	125	9d24 23	inc hl
30	9c71 fe56	vv	cp 'V'	126	9d25 18e6	jr dls
31	9c73 2005		jr nz,bb	127	9d27 af	xor a
32	9c75 3ec9		ld a,#c9	128	9d28 32069d	ld (kres),a
33	9c77 32049e		ld (keyp),a	129	9d2b 32079d	ld (kres+1),a
34	9c7a fe42	bb	cp 'B'	130	9d2e c9	ret
35	9c7c 2005		jr nz,nn	131	9d2f	
36	9c7e 3ec9		ld a,#c9	132	9d2f cd069d	zobr call kres
37	9c80 32109e		ld (sync),a	133	9d32 11049f	ld de,vzor2
38	9c83 fe4e	nn	cp 'N'	134	9d35 2108a0	ld hl,vzor1
39	9c85 2009		jr nz,mm	135	9d38 010001	ld bc,256
40	9c87 3e38		ld a,#38	136	9d3b edb0	ldir
41	9c89 321f9e		ld (updo),a	137	9d3d cd069d	call kres
42	9c8c af		xor a	138	9d40 c9	ret
43	9c8d 32109e		ld (sync),a	139	9d41	
44	9c90 fe4d	mm	cp 'M'	140	9d41 21d05c	data ld hl,23760
45	9c92 2009		jr nz,ww	141	9d44 3ee4	dtr1 ld a,228
46	9c94 3e30		ld a,#30	142	9d46 23	dtr inc hl
47	9c96 321f9e		ld (updo),a	143	9d47 be	cp (hl)
48	9c99 af		xor a	144	9d48 20fc	jr nz,dtr
49	9c9a 32109e		ld (sync),a	145	9d4a 23	inc hl
50	9c9d fe57	ww	cp 'W'	146	9d4b 3e22	ld a,""
51	9c9f 20af		jr nz,scan	147	9d4d be	cp (hl)
52	9ca1 fb		ei	148	9d4e 20f4	jr nz,dtr1
53	9ca2 c9		ret	149	9d50 23	inc hl
54	9ca3			150	9d51 222b9e	ld (dtad),hl
55	9ca3		ivyber casovacich podprogramov	151	9d54 c9	ret
56	9ca3		ivykreslenie sietky	152	9d55	
57	9ca3		isynchronizacia vzorkovania	153	9d55	irutina vypisu na screen
58	9ca3		ivzorkovanie	154	9d55 0600	prnt ld b,0
59	9ca3 fe30	vzor	cp '0'	155	9d57 0e00	prnt2 ld c,0
60	9ca5 28a9		jr z,scan	156	9d59 dd7e00	prnt1 ld a,(ix+0)
61	9ca7 fe39		cp '9'	157	9d5c fe22	cp ""
62	9ca9 28a5		jr z,scan	158	9d5e c8	ret z
63	9cab f5		push af	159	9d5f fe2c	cp ','
64	9cac cd8a9d		call clear	160	9d61 2005	jr nz,prnt
65	9caf f1		pop af	161	9d63 04	inc b
66	9cb0 d631		sub #31	162	9d64 dd23	inc ix
67	9cb2 87		add a,a	163	9d66 18ef	jr prnt2
68	9cb3 212d9e		ld hl,ftab	164	9d68 c5	print push bc
69	9cb6 1600		ld d,0	165	9d69 11003c	ld de,#3c00
70	9cb8 5f		ld e,a	166	9d6c 87	add a,a
71	9cb9 d5		push de	167	9d6d 6f	ld l,a
72	9cba 19		add hl,de	168	9d6e 63	ld h,e
73	9cbb 5e		ld e,(hl)	169	9d6f 29	add hl,hl
74	9cbc 23		inc hl	170	9d70 29	add hl,hl
75	9cbd 56		ld d,(hl)	171	9d71 19	add hl,de
76	9cbe ed53e29c		ld (cas+1),de	172	9d72 eb	ex de,hl
77	9cc2 d1		pop de	173	9d73 78	ld a,b
78	9cc3 213d9e		ld hl,ztab	174	9d74 42	ld b,d
79	9cc6 19		add hl,de	175	9d75 cd9e0e	call #0e9e
80	9cc7 5e		ld e,(hl)	176	9d78 50	ld d,b
81	9cc8 23		inc hl	177	9d79 0600	ld b,#00
82	9cc9 56		ld d,(hl)	178	9d7b 09	add hl,bc
83	9cca ed53d39c		ld (text+2),de	179	9d7c 0608	ld b,#08
84	9cce cdb69d		call gridd	180	9d7e 1a	slc ld a,(de)
85	9cd1 dd21999e	text	ld ix,zf680	181	9d7f 77	ld (hl),a
86	9cd5 cd559d		call prnt	182	9d80 1c	inc e
87	9cd8 2108a0	samp	ld hl,vzor1	183	9d81 24	inc h
88	9cdb 013f00	port3	ld bc,#3f	184	9d82 10fa	djnz slc
89	9cde cd109e		call sync	185	9d84 c1	pop bc
90	9ce1 cda59e	cas	call f680	186	9d85 0c	inc c
91	9ce4 cd2f9d		call zobr	187	9d86 dd23	inc ix
92	9ce7 cd049e		call keyp	188	9d88 18cf	jr prnt1
93	9cea cd229e		call qpress	189	9d8a	
94	9ced 38e9		jr c,samp	190	9d8a 210040	clear ld hl,#4000
95	9cef cdad9d		call cop	191	9d8d 110140	ld de,#4001
96	9cf2 cd8a9d		call clear	192	9d90 010018	ld bc,#1800
97	9cf5 3e18		ld a,#18	193	9d93 75	ld (hl),1
98	9cf7 32069d		ld (kres),a	194	9d94 edb0	ldir
99	9cfa 3e1f		ld a,#1f	195	9d96 3a8d5c	ld a,(#5c8d)
100	9cfc 32079d		ld (kres+1),a	196	9d99 77	ld (hl),a
101	9cff dd2a2b9e		ld ix,(dtad)	197	9d9a 01ff02	ld bc,#02ff
102	9d03 c34d9c		jp scan1	198	9d9d edb0	ldir
103	9d06			199	9d9f c9	ret
104	9d06		ivykreslenie priebehu	200	9da0	
105	9d06 181f	kres	jr prvý	201	9da0 3a039f	vstup ld a,(brana)
106	9d08 21049f		ld hl,vzor2	202	9da3 32169e	ld (port1+1),a
107	9d0b 1600		ld d,0	203	9da6 321c9e	ld (port2+1),a
108	9d0d 7e	dls	ld a,(hl)	204	9da9 32dc9c	ld (port3+1),a
109	9d0e e5		push hl	205	9dac c9	ret

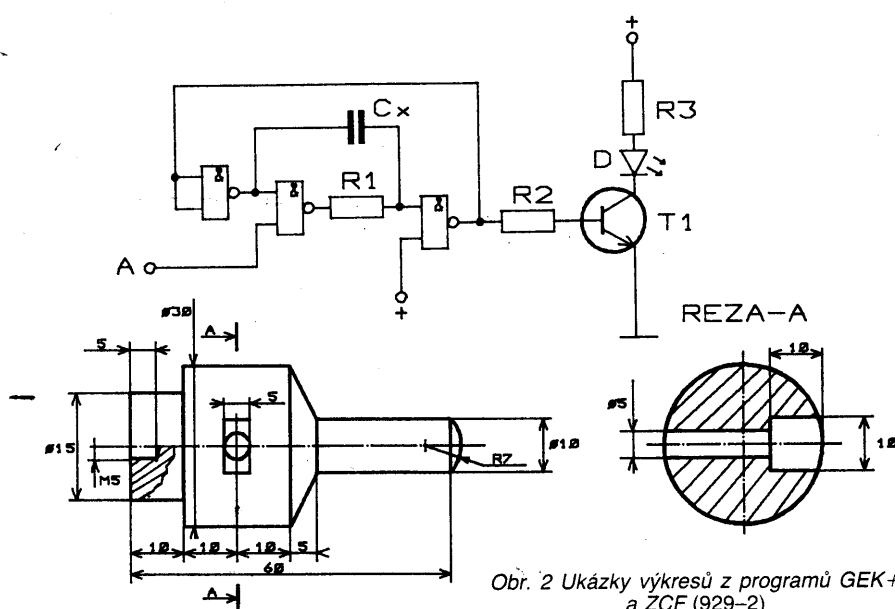


206	9dad				299	9e4b 999e			dw	zf680
207	9dad 3edf	cop	ld	a, #df	300	9e4d 3220732f	zf01	db	'2 s/diel"	
208	9daf dbfe		in	a, (254)		6469656c				
209	9db1 1f		rra			22				
210	9db2 d2b59d		jp	nc, obsl	301	9e56 32303020	zf1	db	'200 ms/diel"	
211	9db5 c9	obs1	ret			6d732f64				
212	9db6					69656c22				
213	9db6	ipodprogram	pre	sietku	302	9e62 3430206d	zf5	db	'40 ms/diel"	
214	9db6 00	gridd	nop			732f6469				
215	9db7 af		xor	a		656c22				
216	9db8 0e00	pkf	ld	c, #00	303	9e6d 38206d73	zf25	db	'8 ms/diel"	
217	9dba f5		push	af		2f646965				
218	9dbb cdb022		call	#22b0		6c22				
219	9dbe 36cc		ld	(hl), #cc	304	9e77 32206d73	zf100	db	'2 ms/diel"	
220	9dc0 e5		push	hl		2f646965				
221	9dc1 d1		pop	de		6c22				
222	9dc2 13		inc	de	305	9e81 302e3820	zf250	db	'0.8 ms/diel"	
223	9dc3 011f00		ld	bc, #1f		6d732f64				
224	9dc6 edb0		ldir			69656c22				
225	9dc8 f1		pop	af	306	9e8d 302e3420	zf500	db	'0.4 ms/diel"	
226	9dc9 c613		add	a, #13		6d732f64				
227	9dcb fed1		cp	#d1		69656c22				
228	9dcd 20e9		jr	nz, pkf	307	9e99 302e3320	zf680	db	'0.3 ms/diel"	
229	9dcf 0e00		ld	c, #00		6d732f64				
230	9dd1 3ebf	tt	ld	a, #bf		69656c22				
231	9dd3 cb47	sm	bit	0, a	308	9ea5			icasovacie rutiny	
232	9dd5 2810		jr	z, idr	309	9ea5 edb2	f680		inir	
233	9dd7 f5		push	af	310	9ea7 c9			ret	
234	9dd8 c5		push	bc	311	9ea8 eda2	f500		ini	
235	9dd9 cdb022		call	#22b0	312	9eaa 20fc			jr	nz, #fc
236	9ddc 3c		inc	a	313	9eac c9			ret	
237	9ddd 47		ld	b, a	314	9ead eda2	f250		ini	
238	9dde 3e01		ld	a, #01	315	9eaf f5			push	af
239	9de0 0f	lp1	rrca		316	9eb0 c600			add	a, #00
240	9de1 10fd		djnz	lp1	317	9eb2 f1			pop	af
241	9de3 ae		xor	(hl)	318	9eb3 20f8			jr	nz, #f8
242	9de4 77		ld	(hl), a	319	9eb5 c9			ret	
243	9de5 c1		pop	bc	320	9eb6 eda2	f100		ini	
244	9de6 f1		pop	af	321	9eb8 f5			push	af
245	9de7 3d	idr	dec	a	322	9eb9 c5			push	bc
246	9de8 20e9		jr	nz, sm	323	9eba 0605			ld	b, #05
247	9dea 3e33		ld	a, #33	324	9ebc 10fe			djnz	#fe
248	9dec 89		adc	a, c	325	9ebe c1			pop	bc
249	9ded d8		ret	c	326	9ebf f1			pop	af
250	9dee 4f		ld	c, a	327	9ec0 20f4			jr	nz, #f4
251	9def 18e0		jr	tt	328	9ec2 c9			ret	
252	9df1				329	9ec3 eda2	f25		ini	
253	9df1	irutina nasobenja 3/4			330	9ec5 f5			push	af
254	9df1 2600	prep	ld	h, #00	331	9ec6 c5			push	bc
255	9df3 6f		ld	l, a	332	9ec7 0625			ld	b, #25
256	9df4 54		ld	d, h	333	9ec9 10fe			djnz	#fe
257	9df5 5d		ld	e, l	334	9ecb c1			pop	bc
258	9df6 29		add	hl, hl	335	9ecc f1			pop	af
259	9df7 19		add	hl, de	336	9ecd 20f4			jr	nz, #f4
260	9df8 cb3c		srl	h	337	9ecf c9			ret	
261	9dfa cb1d		rr	l	338	9ed0 eda2	f5		ini	
262	9dfc cb3c		srl	h	339	9ed2 f5			push	af
263	9dfe cb1d		rr	l	340	9ed3 c5			push	bc
264	9e00 3ebf		ld	a, #bf	341	9ed4 06cd			ld	b, #cd
265	9e02 95		sub	l	342	9ed6 10fe			djnz	#fe
266	9e03 c9		ret		343	9ed8 c1			pop	bc
267	9e04				344	9ed9 f1			pop	af
268	9e04	ivstup z klavesnice			345	9eda 20f4			jr	nz, #f4
269	9e04 00	keyp	nop		346	9edc c9			ret	
270	9e05 cd8e02	keys	call	#028e	347	9edd eda2	f1		ini	
271	9e08 20fb		jr	nz, keys	348	9edf f5			push	af
272	9e0a cd1e03		call	#031e	349	9ee0 c5			push	bc
273	9e0d 30f6		jr	nc, keys	350	9ee1 0600			ld	b, #00
274	9e0f c9		ret		351	9ee3 10fe			djnz	#fe
275	9e10	isynchronizacia			352	9ee5 10fe			djnz	#fe
276	9e10 00	sync	nop		353	9ee7 10fe			djnz	#fe
277	9e11 cd229e		call	qpress	354	9ee9 10fe			djnz	#fe
278	9e14 d0		ret	nc	355	9eeb c1			pop	bc
279	9e15 db3f	port1	in	a, (63)	356	9eec f1			pop	af
280	9e17 fe7f	sync1	cp	127	357	9eed 20ee			jr	nz, #ee
281	9e19 20f5		jr	nz, sync	358	9eef c9			ret	
282	9e1b db3f	port2	in	a, (63)	359	9ef0 eda2	f01		ini	
283	9e1d fe7f		cp	127	360	9ef2 f5			push	af
284	9e1f 38ef	updo	jr	c, sync	361	9ef3 c5			push	bc
285	9e21 c9		ret		362	9ef4 0629			ld	b, #29
286	9e22				363	9ef6 c5			push	bc
287	9e22 cdad9d	qpress	call	cop	364	9ef7 0600			ld	b, #00
288	9e25 3efb		ld	a, #fb	365	9ef9 10fe			djnz	#fe
289	9e27 dbfe		in	a, (254)	366	9efb c1			pop	bc
290	9e29 1f		rra		367	9efc 10f8			djnz	#f8
291	9e2a c9		ret		368	9efe c1			pop	bc
292	9e2b	itabulky			369	9eff f1			pop	af
293	9e2b 0000	dtad	db	0, 0	370	9f00 20ee			jr	nz, #ee
294	9e2d f09edd9e	ftab	dw	f01, f1, f5	371	9f02 c9			ret	
	d09e				372	9f03 fe			brana	db 254
295	9e33 c39eb69e		dw	f25, f100	373	9f04			ibuffer pre	vzorky
296	9e37 ad9ea89e		dw	f250, f500, f680	374	9f04			vzor2	ds 260
	a59e				375	a008			vzor1	ds 260
297	9e3d 4d9e569e	ztab	dw	zf01, zf1, zf5, zf25	376	a10c				
	629e6d9e				377	a10c				
298	9e45 779e819e		dw	zf100, zf250, zf500					end	
	8d9e									

# PROGRAM PRO KRESLENÍ GEK+

**Jan Věříš,** *Leninova 268, 533 41 Lázně Bohdaneč*

V poslední době se začal mezi vlastníky domácích počítačů rozšiřovat zapisovač jako periferní zařízení. Rozlišovací schopnost běžných zapisovačů je podstatně větší než rozlišení na obrazovce počítače. Většina kreslicích programů pro ZX Spectrum však provádí kresbu přímo na obrazovku a nevytváří přitom žádný záznam o vzniku kresby. V tomto případě pak slouží zapisovač pouze jako pomalejší tiskárna a jeho hlavní přednosti nejsou využity. Aby bylo možno plně využít velkou rozlišovací schopnost zapisovače, je nutné, aby program zaznamenával kreslené objekty v tzv. vektorové formě do paměti (tj. zaznamenával pouze souřadnice koncových bodů a některé další důležité parametry s co největší přesností, podle kterých lze kresbu zpětně zrekonstruovat. Výsledný vektorový soubor lze pak vykreslit na souřadnicovém zapisovači s maximální možnou přesností.



Obr. 2 Ukázky výkresů z programů GEK+  
a ZCF (929-2)

Nyní můžeme posunovat barevným kurzorem v pravé části obrazovky po jednotlivých příkazech menu pomocí kláves Q, A – pomalý posun a O, P – posun po stránkách. Vybranou funkci potvrdíme klávesou Z nebo M. Pomocí klávesy X můžeme kdykoliv opustit toto menu bez provedení příkazu. Při častém užívání některých funkcí by stále hledání v menu zbytečně zpomalovalo práci, proto je možné předdefinovat tři funkce na tlačítka Z, N a M. Tyto funkce jsou pak kdykoliv přístupné stiskem příslušného tlačítka bez nutnosti jejich výběru z menu. Postup je následující: nastavíme kurzor na vybranou funkci a stiskneme N (=DEFINE), potom stiskneme jednu z kláves Z, N nebo M. Nyní již je vybraná funkce přístupná přes vybranou klávesu přímo bez volání menu. Jméno definované funkce se také zobrazí v příslušné kolonce dialogového řádku.

### 3. Operace s bloky

## 1. Základní filozofie programu GEK+

Kreslicí plocha programu GEK+ má rozměr 20000×20000 základních jednotek (1 základní jednotka = 1 pixel při zobrazení ZOOM 1:1 viz dále). Na této pracovní ploše lze umísťovat základní objekty – čáry (LINE), kruhové oblouky (ARC), a text o maximální délce 5 znaků (TEXT).

Jedna kreslicí plocha se nazývá blok. Lze otevřít (OPEN) až 60 nezávislých kreslicích ploch – bloků. Každý otevřený blok je v podstatě dalším základním objektem, který lze tedy stejně jako čáru, text, oblouk umístit kamkoliv do jiného bloku. Umisťování bloků do sebe je omezeno pouze tak, aby nevznikla rekurze – tedy nelze vložit blok do sebe samého.

Obrazovka počítače je při běhu programu GEK + rozdělena na tři samostatné části. V pravé části je trvale zobrazena nabídka – menu přístupných příkazů. Ve spodní části obrazovky je tzv. dialogová řádka, na které program zobrazuje jednoduchou nápovědu podle okamžité situace. Na zbytku obrazovky je zobrazen výřez z kreslicí plochy právě aktivního bloku s ohledem na zvolení zvětšení (ZOOM) a posun (PAN).

## 2. Ovládání programu

Program se ovládá pomocí osmi tlačítek klávesnice. Čtyři pro pohyb kurzoru (Q nahoru, A dolů, O vlevo, P vpravo). Čtyři tlačítka Z, X, M, N slouží pro výběr variant.

2, X, W, N sledují pro výčet varianty.

• Po nahrání programu se zobrazí kurzor v podobě křížku uprostřed obrazovky. Při pohybu se jeho rychlost po skocích postupně zvyšuje. Po dosažení okraje obrazovky se kurzor zastaví. V horní části obrazovky se průběžně tiskne x-ová a y-ová vzdálenost kurzoru od bodu o souřadnicích 0,0 v milimetrech (1 základní jednotka = 0,125 mm, což odpovídá nejmenšímu kroku zapisovací).

Dialogová řádka je pomyslně rozdělena na čtyři části; každá část odpovídá jednomu tlačítku pro výběr varianty. Zleva to tedy jsou Z, X, N a M. Po spuštění programu tato řádka vypadá takto:

*ERASE*      *MENU*      *TEXT*      *LINE*

Tlačítkem Z tedy vyvoláme funkci *ERASE*, tlačítkem N funkci *TEXT* a tlačítkem M funkci *LINE*. Stiskneme-li klávesu X, dostaneme se do menu. V dialogové řádce se objeví tato zpráva:

SELECT CREATE DEFINE SELECT

### 3.1 OPEN

Po spuštění programu není otevřen žádný blok – neexistuje tedy kreslicí plocha, do které bychom mohli umisťovat objekty. Pro otevření nové kreslicí plochy slouží funkce OPEN. Aktivujeme-li tuto funkci, program nejprve kontroluje, zda lze otevřít další blok (počet bloků je maximálně 60). V případě, že je otevřeno již 60 bloků, program ohlásí chybu „Too many blocks“. Je-li vše v pořádku, program požádá zadání šestiznakového jména otevíraného bloku. Po napsání jména stiskneme ENTER. V této chvíli je nový blok otevřen a jeho jméno se stalo součástí menu.

### 3.2 CURR

V jednom okamžiku lze samozřejmě kreslit pouze do jednoho tzv. aktivního bloku. Chceme-li kreslit či provádět změny v jiném již otevřeném bloku, vyvoláme funkci CURR. Nyní se v pravé části obrazovky zobrazí pouze poslední část menu se jmény otevřených bloků a v dialogovém řádku se vypíše nápověda:

*Backgr. Current block*

Vybereme blok, který se má stát aktivním blokem a potvrdíme volbu klávesou X, N nebo M. Stiskneme-li klávesu Z, vybraný blok se vykreslí méně výrazně jako pozadí pod následně vybraný aktivní blok. Toto je výhodné např. při kreslení dvoustranných desek plošných spojů apod.

Pozn.:

1. Blok otevřený funkcí OPEN se zároveň stává i aktivním blokem a taktéž základním objektem (viz kapitola 2).

II. Vybereme-li kurzorem prázdný řádek v nabídce, program se chová tak, jako bychom vybrali funkci popř. blok, jehož jméno je zobrazeno první.

## 4. Kreslení objektů

V programu GEK+ existují čtyři druhy základních objektů. Jsou to čáry (příkaz *LINE*), kruhové oblouky (*ARC*), text (*TEXT*). Každý otevřený blok se jeví z hlediska druhých bloků také jako objekt – tj. lze ho umístit na libovolné místo jiného bloku a to i několikrát.

### 4.1 Čáry – příkaz *LINE*

Kurzor umístíme do počátečního bodu čáry a vyvoláme funkci *LINE*, v dialogovém řádku se objeví zpráva *To point*, kurzor tedy nastavíme na koncový bod čáry a potvrdíme (klávesy Z, X, N, M). Na obrazovce se nakreslí rovný úsek čáry specifikované tloušťky (viz kapitola 5.4), v dialogové řádce se zobrazí:

*END To point*

Klávesou Z tedy můžeme funkci *LINE* opustit nebo lze nastavit kurzor do dalšího bodu a potvrdit (X, N nebo M). Tím je možné kreslit další úseky lomené čáry.

### 4.2 Kruhové oblouky – příkaz *ARC*

Vyvoláme funkci *ARC*, v dialogové řádce se objeví: *Center point*, nastavíme tedy kurzor na střed kruhového oblouku a potvrdíme (Z, X, N nebo M), v dialogové řádce se objeví: *Start point*, nastavíme kurzor na počáteční bod oblouku a potvrdíme. Nyní program požaduje zadání úhlu opsaného obloukem ve stupních (kladný údaj – oblouk bude pokračovat od počátečního bodu proti směru hodinových ručiček, záporný údaj – po směru hodinových ručiček). Zadaný číselný údaj potvrdíme stiskem klávesy *ENTER*.

### 4.3 Text příkaz *TEXT*

Kurzor nastavíme do bodu, kde se má nacházet levý dolní roh řádky textu. Vyvoláme funkci *TEXT*. Program požaduje zadání textového řetězce o délce právě 5 znaků. Pokud je text kratší, je nutné jej doplnit mezerami !! Text potvrdíme *ENTER*. Nyní zadáme výšku textu v milimetrech a opět potvrdíme *ENTER*. Pro ušetření místa v paměti a pro zrychlení programu se na obrazovce nevytiskne text, ale pouze rámeček, který respektuje délku a rozměry textu.

### 4.4 Vkládání bloků

Umístíme kurzor na místo, kde se má nacházet bod o souřadnicích 0,0 ve vkládaném bloku. Nyní v menu nastavíme kurzor na jméno bloku, jež chceme vložit do právě aktivního bloku a potvrdíme stejně jako jakoukoliv jinou funkci (Z nebo M).

*Pozn.:*

I. Jméno bloku lze samozřejmě stejně jako jinou funkci předdefinovat na klávesy Z, N nebo M.

II. Program nepřipouští rekurzivní vkládání bloků do sebe. Nelze tedy vložit blok 1 do bloku 1 – tomu program zabrání, ale je-li blok 1 vložen do bloku 2 a dojde k pokusu o vložení bloku 2 do bloku 1, program to nerozpozná a havaruje !!

## 5. Opravy výkresu

I při nejpečlivější práci se můžeme zmýlit. Pro opravy výkresu slouží v programu GEK+ tři funkce: vymazávání (*ERASE*), posun (*MOVE*), a kopírování (*COPY*).

### 5.1 Vymazávání – příkaz *ERASE*

Touto funkcí lze vymazat všechny objekty, které alespoň částečně zasahují do definované obdélníkové oblasti. Pokud je do právě aktivního bloku vložen jiný blok, je tento chápán jako jediný objekt, tedy – zasahuje-li do vymezené oblasti alespoň jeden objekt z vloženého bloku, bude celý tento vložený blok vymazán. Po vyvolání funkce *ERASE* označíme nejprve levý dolní roh oblasti, kterou chceme vymazat. Potom označíme pravý horní roh této oblasti. Po potvrzení obou bodů program jakoby „v duchu“ musí nakreslit znova všechny objekty a vybrané odstranit z paměti. Zdánlivě se chvíli nic neděje. Když je program hotov, vymaže obrazovku a nakreslí znova aktivní blok již bez vymazaných objektů. Pozor! V případě složitější kresby může být tato činnost velmi časově náročná.

*Pozn.:* Při znovuvykreslení aktivního bloku funkcí *ERASE* se již nevykreslí případné pozadí (viz kapitola 3. 2).

### 5.2 Přesun *MOVE*

Tato funkce umožňuje přesunout výřez právě aktivního bloku na jiné místo tohoto bloku. Postup je obdobný jako u funkce *ERASE*. Po zadání přesouvané oblasti program chvíli „přemýšlí“, poté jsme dotázáni na polohu bodu, do kterého se přesune původně označený levý dolní roh přesouvané oblasti. Další činnost programu je obdobná funkci *ERASE*.

*Pozn.:* viz 5.1

### 5.3 Kopírování *COPY*

Tato funkce okopíruje objekty z definované oblasti na jiné místo. Postup je obdobný jako u funkce *MOVE*.

### 5.4 Příkaz *LINE TYPE (LTYPE)*

Umožňuje zadat počet paralelně jdoucích čar a jejich vzdálenost, které se vykreslí při použití *LINE* nebo *ARC*. Pomocí tohoto příkazu tedy lze kreslit různě tlusté čáry.

Nejprve zadáme počet paralelně jdoucích čar (musí být větší nebo roven jedné) a potom vzdálenost mezi jednotlivými čarami v základních jednotkách (1 základní jednotka = 1 pixel při zobrazení *ZOOM 1:1* viz níže).

Definovaný typ čáry platí až do dalšího předdefinování. Po nahrání programu je automaticky nastaven typ: počet čar = 1, vzdálenost = 0 (pokud je počet čar roven jedné, nemá údaj vzdálenosti žádný význam).

## 6. Skupina příkazů pro řízení zobrazování

Protože kreslicí plocha programu GEK+ je v porovnání s obrazovkou počítače daleko větší, není vždy možné zobrazit na obrazovce celý výkres s dostatečnou přesností. Na obrazovce je vidět pouze výřez okolí definovaného bodu kreslicí plochy (po spuštění je to okolí bodu 0,0) v definovaném zmenšení. Pomocí funkce *ZOOM* je možné měnit zmenšení kresby, funkcí *PAN* lze definovat bod, jehož okolí chceme zobrazovat.

### 6.1 Příkaz *ZOOM*

Umožňuje měnit měřítko zobrazení na obrazovce. Souřadnice objektů jsou v paměti počítače uloženy s konečnou přesností. Zadáme-li měřítko 1:1, bude vzdálenost dvou sousedních pixelů na obrazovce zároveň nejmenší možnou vzdáleností, kterou může program GEK+ rozlišit. Tato vzdálenost také souhlasí s rozlišením zapisovače, které je 0,125 mm. Navývá se základní jednotka.

Po vyvolání příkazu *ZOOM* se v dialogové řádce zobrazí výzva: *Scale 1:* a program čeká na zadání čísla z intervalu 1 až 99

včetně, které říká, kolik základních jednotek připadne na jeden pixel. Jsou povolena i desetinná čísla. Po zadání hodnoty program znovuvykreslí výkres v zadaném měřítku.

### 6.2 Příkaz *PAN*

Na obrazovce je zobrazena část kreslicí plochy v okolí bodu 0,0. Chceme-li pracovat na jiné části výkresu, která je momentálně mimo rámec obrazovky, použijeme příkaz *PAN*. Po jeho vyvolání program vyžaduje zadání souřadnic bodu, který má být ve středu obrazovky. Po zadání souřadnic program smaže obrazovku a znovuvykreslí okolí definovaného bodu. Při kreslení se samozřejmě respektuje dříve zadané zmenšení.

## 7. Příkazy pro spolupráci s magnetofonem

### 7.1 *SAVE*

Pomocí tohoto příkazu je možno uschovat na magnetofonový pásek jeden či několik bloků.

Po vyvolání tohoto příkazu program nejprve požaduje zadání prvního nahrávaného bloku. V nabídce tedy vybereme kurzorem první blok, který chceme nahrát a potvrdíme klávesou Z, X, N nebo M. Stejným způsobem označíme i poslední nahrávaný blok. Nyní program požaduje zadání jména nahrávky. Tento název musí mít přesně 5 znaků! Po odeslání jména (*ENTER*) spustíme magnetofon. Nahrávají se postupně tři soubory, které nesou všechny potřebné informace o označených blocích.

### 7.2 Příkaz *LOAD*

Pomocí tohoto příkazu lze nahrát do paměti počítače bloky dříve uchované příkazem *SAVE*. Po vyvolání příkazu *LOAD* zadáme pětiznakové jméno, které jsme použili při ukládání bloků na pásek pomocí *SAVE*. Poté spustíme magnetofon. Program automaticky řídí nahrání všech tří částí záznamu. Nahráný blok (bloky) se připojí za již otevřené bloky. Jestliže by při připojování bloků příkazem *LOAD* hrozilo překročení vymezené kapacity paměti, vypíše se hlášení „*No room for file*“ a volba je zrušena.

## 8. Další důležité informace

8.1 Zatímco u klasických kreslicích programů je omezena pracovní plocha počtem zobrazitelných bodů na obrazovce, u programu GEK+ je kreslicí plocha prakticky neomezená (20000×20000 základních jednotek odpovídá přibližně rozměru výkresu 2,5×2,5 metrů při rozlišení 0,125 milimetrů), je však omezen celkový počet objektů, které je možno do kreslicí plochy (kreslicích ploch) umístit. Pro záznam objektů je vymezeno přibližně 8 kB operační paměti, což odpovídá přibližně 1500 úsekům rovných čar či 700 kruhovým obloukům. Pokud se právě zadávaný objekt již nevejde do paměti, program vypíše zprávu „*Memory full*“. Objekt se sice nakreslí na obrazovku, ale do paměti se již nezaznamená.

8.2 Vložíme-li blok jako objekt do jiného bloku, nezaznamenává se do paměti jméno tohoto bloku, ale pouze jeho pořadové číslo. To může ve spojení s příkazy *SAVE* a *LOAD* činit určité potíže.

*Příklad:*

Mějme nahráný na páse soubor s informacemi o těchto třech blocích:

1. **CTVER** obsahuje čtverec vytvořený ze čtyř úseků rovné čáry,
2. **KOLO** obsahuje kruh vytvořený příkazem *ARC*,
3. **CTVK** obsahuje vnořené bloky **CTVER**. a **KOLO**.

Po nahrání programu GEK+ otevřeme příkazem *OPEN* první blok a nazveme jej *KRIZEK*, bude obsahovat např. křížek složený ze dvou rovných čar. Nyní použijeme příkaz *LOAD* a přihrájeme výše uvedený soubor bloků. Hlavní menu potom může vypadat například takto:

KRIZEK  
CTVER  
KOLO  
CTVK

Podíváme-li se nyní příkazem *CURR* na bloky *KRIZEK*, *CTVER*, a *KOLO*, jsou v pořádku. Podíváme-li se však na blok *CTVK*, vidíme křížek a čtverec, ač zde byl původně čtverec a kruh. Blok *CTVK* obsahuje první a druhý blok – původně to byly bloky *CTVER*, a *KOLO*, nyní je však první blok *KRIZEK* a druhý *CTVER*, a tak se také zobrazí.

## 9. Popis vnitřní struktury programu

Zdrojový text programu GEK+ byl napsán v jazyce Pascal a přeložen kompilátorem hp113, což je upravená verze kompilátoru HP4T1.6 fy HiSoft doplněná o některé grafické procedury.

Program lze pro zjednodušení rozdělit na několik částí: blok procedur pro ovládání datového souboru s uloženými objekty, blok funkcí pro kreslení objektů na obrazovku, blok procedur, které provádějí uživatelem vybrané funkce, část, která řídí komunikaci s uživatelem (ovládání menu, kurzoru atd.) a nakonec část programu, která řídí součinnost všech těchto částí.

Funkce pro kreslení objektů:

- LINETL** vykreslí čáru dle zadaných parametrů a platného měřítka a posunutí. Výstupem funkce je log. hodnota typu Boolean, která říká, zda nakreslená čára zasahuje do definovaného okénka (používá se při *MOVE*, *COPY* a *ERASE*).
- ARCKRES** kreslení kruhového oblouku podle zadaného středu, počátku, opsaného úhlu a typu čáry. Ostatní viz *LINETL*.
- TEXTKRES** kreslení rámečku namísto textu.
- KRESLI** kreslení všech objektů, které obsahuje parametry specifikovaný blok. Funkce rekurzivně volá sama sebe, je-li do kresleného bloku vložen další blok. Ostatní viz *LINETL*.

Procedury pro ovládání datového souboru

Protože použitý překladač neumožňuje vytvořit vyhovující strukturu dynamických proměnných pro ukládání údajů o kreslených objektech, bylo nutné vytvořit procedury, které udržují speciální datové pole v paměti pro tyto objekty. Toto pole je tvořeno jednak samostatným úsekem paměti o délce asi 8 kB a dále skupinou ukazatelů na začátku jednotlivých podúseků v datovém poli. Struktura datového pole je na *obr. 1*.

Všechny ukazatele jsou soustředěny v poli *UK*, kromě toho jsou v tomto poli též uloženy údaje o počtu objektů v jednotlivých oblastech. Práci s informacemi o objektech umožňují tyto procedury a funkce:

**POCETPRVKU** vrací počet objektů uložených v specifikované oblasti.

- NPRVEK** vydá n-tý prvek ze zadané oblasti.
- VLOZ** vloží prvek do zadané oblasti.
- SMAZ** smaže vybraný prvek v zadané oblasti.
- ZMEN** změní údaje u specifikovaného prvku v zadané oblasti.

*Zadaná oblast* – je to oblast, ve které jsou uloženy informace o objektech stejného typu. Tato oblast je definována číslem bloku a typem oblasti (*LINE*, *ARC* ...), což jsou parametry každé procedury.

Komunikace s uživatelem

Funkce **CPRIK** tato funkce řídí pohyb kurzoru, výběr z menu a předdefinování uživatelských kláves. Vrací číslo příkazu, který byl vybrán uživatelem.

Funkce **POHYBKUR** řídí pohyb kurzoru po obrazovce, přepočít souřadnic, zobrazování souřadnic, osového kříže apod. Vrací znak odpovídající stisknuté klávese (kromě Q, A, O a P).

Procedury provádějící příkazy vyvolané z menu uživatelem: **PROPEN**, **PRCURR**, **PRERASE**, **PRMOVE**, **PRCOPY**, **PRLINE**, **PRZOOM**, **PRPAN**, **PRSAVE**, **PRLOAD**, **PRLINE**, **PRARC**, **PRTEXT**, **PRBLOK**.

Řízení běhu programu

Vlastní prováděcí program nejprve volá funkci *CPRIK* a podle čísla požadovaného příkazu vyvolá příslušnou proceduru (viz procedury provádějící příkazy ...). Po ukončení vyvolané procedury se činnost opakuje.

Programy GEK+ a ZCF jsou napsány v jazyce Pascal. Program GEK+ byl přeložen upraveným kompilátorem hp113 na počítači Spectrum 80 kB. Na Spectru 48 kB nelze program vzhledem k délce zdrojového textu a výsledného kódu přeložit. Program ZCF lze s jistými omezeními přeložit i v paměti 48 kB. Použitý kompilátor hp113 má zabudovány některé nestandardní procedury, které by bylo nutné při použití jiného překladače doprogramovat:

- AT(R,S:INTEGER)** přesune pozici pro tisk na řádek R a sloupec S.
- PLOT(X,Y,A:INTEGER)** vytiskne bod o souřadnicích X a Y, parametr A specifikuje barvu bodu.

- DRFT** body se automaticky spojují úsečkami.
- DOT** ruší účinek *DRFT*.

**TOUT(Jméno:ARRAY (1..8) OF CHAR; START,DÉLKA:INTEGER)**

nahrává na magnetofon specifikovaný úsek paměti.

**TIN(Jméno:ARRAY (1..8) OF CHAR; START:INTEGER)**

nahrává data z magnetofonu do paměti.

Také v příkazu *WRITE* jsou použity některé nestandardní řídicí kódy:

**CHR(21),CHR(B)** ekvivalent *OVER B* v *BASiC*.

**CHR(17),CHR(B)** ekvivalent *PAPER B* v *BASiC*.

Při přenosu programu na jiný typ počítače by bylo nutné upravit:

- kontrolu souřadnic v procedurách *LINEKRES*, *KRIZW*, *POHYBKUR* podle rozlišovací schopnosti použitého počítače,
- souřadnice použité při volání procedury *AT* podle počtu zobrazitelných řádků a sloupců,
- adresu začátku a konce volné paměti pro data v konstantách *OFFSET* a *TOPMEM*.

Program ZCF je vystavěn kolem ovladače *SUPERPLOT*, který je dodáván se zapisovačem XY 4131. V případě použití jiného zapisovače a tím i ovladače by bylo nutné přeprogramovat procedury, které zajišťují předání parametrů z Pascalu ovladači:

MA, VA, AC, OG, MF, CS, SC, ODVYS5. Podrobně jsou tyto příkazy popsány v příručce k programovému vybavení zapisovače XY 4131.

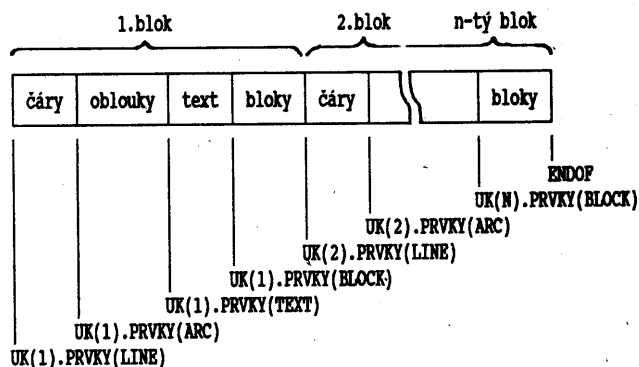
Pokud někdo nebude schopen program sám přeložit, může se obrátit na autora o nahrávku zkompilovaného programu.

## 10. Ná vaznost programu na kreslicí zařízení

Popis programu ZCF

Pro překreslení kresby zaznamenané programem GEK+ na souřadnicovém zapisovači byl napsán další kratší program. Je vystavěn kolem ovladače programu *SUPERPLOT*, který je dodáván společně se zapisovačem XY 4131. Bohužel jsem nikdy neměl možnost vyzkoušet tento program v originální sestavě (používám upravený *SUPERPLOT* pro ovládání zapisovače pouze přes jediný osmibitový port), proto nemohu stoprocentně zaručit bezchybnou spolupráci se zapisovačem XY 4131.

Po nahrání programu příkazem *LOAD* se ZCF automaticky spustí. Nyní musíme do programu nahrát všechny bloky vytvořené



Obr. 1. Struktura datového pole (929-1)

programem GEK+ nutné pro správné zrekonstruování kresby (viz kapitola 8.2). Na dotaz „Name“ napíšeme pětiznakové jméno souboru, který jsme před tím zaznamenali programem GEK+ příkazem SAVE a spustíme magnetofon. Postupně budou nahrány tři bloky dat s potřebnými údaji. Pokud je třeba přehrát ještě další soubor (soubory) jiného jména, stiskneme Y a postup se opakuje až máme všechny potřebné bloky nahrány v paměti.

Nyní program požaduje tyto údaje:

- Scale** měřítko zvětšení kresby (menší než 1... zmenšení, větší než 1... zvětšení).
- Origin X,Y** souřadnice vztažného bodu – na tento bod později nastavíme písátko zapisovače a tím určíme polohu kresby na papíře.
- Orientation** 1... standardní kresba,  
2... kresba bude zrcadlově otočena podle osy X  
3... kresba bude otočena o 90° podle počátku.

Po zadání všech těchto údajů proběhne inicializace zapisovače a pak program čeká, až pomocí kláves 5, 6, 7, 8 příp. +1 pro zrychlený pohyb nastavíme písátko do bodu, jež má na papíře odpovídat výše zadanému vztažnému bodu. Jsme-li hotovi, stiskneme 0. V tomto okamžiku program zobrazí všechny nahrané bloky a jejich pořadová čísla a čeká na zadání počtu bloků, které chceme najednou vykreslit (ve většině případů jsou to jeden nebo dva např. oboustranné desky

plošných spojů). Nyní postupně napíšeme pořadová čísla těchto bloků tak, jak jsou uvedena na obrazovce. Poté program vypíše zprávu „I am drawing“ a kreslí specifikované bloky.

**Pozn.:**

I. Kreslení lze kdykoliv přerušit klávesou SPACE. Po stisku kterékoliv jiné klávesy kreslení pokračuje.

II. Volba Orientation pracuje správně pouze s upraveným ovládačem zapisovače. Je-li nahrán originální ovládač, volba je ignorována.

## 11. Ukázkový soubor PLSSP

Tento soubor nahrajeme do programu GEK+ (případně ZCF) příkazem LOAD (popř. výše uvedeným postupem) a zadáním jména PLSSP. Soubor obsahuje několik bloků:

- MALYB malý pájecí bod Ø 1,25 mm,  
VELKYB velký pájecí bod Ø 3 mm,  
DIL14S patice DIL 14 svisle,  
DIL14V patice DIL 14 vodorovně,  
DIL16S patice DIL 14 svisle,  
DIL16V patice DIL 14 vodorovně,  
OZ patice operačního zesilovače (kulatá),  
X1, X2 pomocné bloky,  
MRIZ5 pomocný rastr 5+5 mm,  
STRSPO příklad plošného spoje vytvořeného na  
STRSOU základě tohoto souboru pájecích bodů a patic.

### Výpis 1. Program GEK+ (929-V1)

```
PROGRAM GEK;
CONST OFFSET=EA60;
TOPMEM=F450;
PBLMAX=80;
POCPRIK=23;
JMCCLKEM=83;
TYPE LINET=RECORD
  X,Y:INTEGER;END;
ARCT=RECORD XC,YC,XS,YS,A,U:INTEGE
R;END;
TEXTT=RECORD VELIKOST:CHAR;XP,YP:I
NTEGER;T:ARRAY[1..5] OF CHAR;END;
OBJT=(LINE,ARC,TEXT,BLOK);
UKAZATELE=ARRAY[1..PBLMAX] OF RE
CORD
RVKY:ARRAY[LINEL..BLOK] OF INTEGER;
OCET:ARRAY[LINEL..BLOK] OF INTEGER;
ND;
BLOKT=RECORD X,Y:INTEGER;CBL:CHAR;
END;
NAMET=ARRAY[1..10] OF CHAR;
VAR X3,Y3,X4,Y4,PANX,PANY,GEL:INTEGER;
SCALE:REAL;
KRES:BOOLEAN;
PLINE:LINET;PARC:ARCT;PTXT:TEXTT;P
BLOK:BLOKT;
UK:UKAZATELE;ENDOF:INTEGER;
XCUR,YCUR,XCURS,YCURS,PBLAKT,MNUA,
MNUR:INTEGER;
RYCHLOST:ARRAY[1..3] OF RECORD
  XRY,YRY,P:INTE
GER;END;
JMENA:ARRAY[1..JMCCLKEM,1..6] OF CH
AR;
DP:ARRAY[1..3] OF INTEGER;
CURBLOK,CURLT:INTEGER;
CISLO:INTEGER;
FUNCTION LINEKRES(X1,Y1,X2,Y2:INTEGER):
BOOLEAN;
VAR M1,M2,XR,YR:REAL;
N,M:INTEGER;
B2:BOOLEAN;
```

```
BEGIN
  XR:=((X1+PANX)*SCALE);YR:=((Y1+PANY)*S
  CALE);
  M1:=((X2+PANX)*SCALE);M2:=((Y2+PANY)*S
  CALE);
  LINEKRES:=FALSE;
  B2:=((XR<0) AND (M1<0)) OR ((XR>206) AN
  D (M1>206));
  B2:=B2 OR ((YR<0) AND (M2<0)) OR ((YR>1
  75) AND (M2>175));
  IF NOT B2 THEN BEGIN
    M:=ROUND(SQRT((X2-X1)*SQRT(Y2-Y1))*SC
    ALE);
    IF M=0 THEN M:=1;
    M1:=(X2-X1)/M*SCALE;M2:=(Y2-Y1)/M*SCA
    L;
    FOR N:=0 TO M DO BEGIN
      X1:=ROUND(XR);Y1:=ROUND(YR);
      IF (X1>=0) AND (X1<=206) AND (Y1>=0) A
      ND (Y1<=175) AND KRES THEN PLOT(X1,Y1,0
      );
      IF (X1>=X3) AND (X1<=X4) AND (Y1>=Y3)
      AND (Y1<=Y4) THEN LINEKRES:=TRUE;
      XR:=(XR+M1);YR:=(YR+M2);
      END;
      END;
      END;
      FUNCTION LINETL(X1,Y1,X2,Y2,A:INTEGER):
      BOOLEAN;
      VAR N,M:INTEGER;M1,M2,XR,YR:REAL;
      BEGIN
        LINEKRES:=FALSE;
        N:=A MOD 100;A:=A DIV 100;
        M1:=SQRT((X2-X1)*SQRT(Y2-Y1));
        IF M1=0 THEN M1:=1;
        M2:=(Y2-Y1)/M1*A*(N-1)/2;
        M1:=(X2-X1)/M1*A*(N-1)/2;
        X2:=ROUND(X2-X1);XR:=(X1-M2);
        Y2:=ROUND(Y2-Y1);YR:=(Y1-M1);
        IF N>1 THEN BEGIN M1:=M1/((N-1)/2);M2:=
        M2/((N-1)/2);END;
        FOR M:=1 TO N DO BEGIN
          X1:=ROUND(XR);Y1:=ROUND(YR);
          IF LINEKRES(X1,Y1,X1+X2,Y1+Y2) THEN L
          INETL:=TRUE;
          XR:=XR+M2;YR:=YR-M1;
          END;
          END;
          FUNCTION ARCKRES(AC:ARCT):BOOLEAN;
```

```
VAR C,PR,PA,DA:REAL;N,M,D,E:INTEGER;
BEGIN
  WITH AC DO BEGIN
    C:=U/1000;
    PR:=SQRT((X2-X1)*SQRT(Y2-Y1));
    N:=ROUND(ABS(PR*SCALE*2*(C/6.28)));
    PA:=ARCTAN((Y2-Y1)/(X2-X1+0.01));
    IF (X2-X1)<0 THEN PA:=PA+3.1415;
    ARCKRES:=FALSE;
    IF N=0 THEN ARCKRES:=LINETL(XS,YS,XS,
    YS,A)
    ELSE BEGIN
      DA:=C/N;
      FOR M:=1 TO N DO BEGIN
        D:=ROUND(PR*COS(PA+DA)+XC);E:=ROUND
        (PR*SIN(PA+DA)+YC);
        IF LINETL(ROUND(XC+PR*COS(PA)),ROUND(YC
        +PR*SIN(PA)),D,E,A) THEN ARCKRES:=TRUE;
        PA:=PA+DA;
        END;
        END;
        END;
        END;
        FUNCTION TEXTKRES(TXT:TEXTT):BOOLEAN;
        VAR SKV,SKD,M:INTEGER;
        BEGIN
          WITH TXT DO BEGIN
            SKV:=ORD(VELIKOST)*4;SKD:=0;
            FOR M:=1 TO 5 DO BEGIN
              IF (T[M]<>' ') AND (T[M]<>CHR(13)) T
              HEN SKD:=SKD+SKV;END;
              TEXTKRES:=FALSE;
              IF LINEKRES(XP,YP,XP+SKD,YP) THEN TEX
              TKRES:=TRUE;
              IF LINEKRES(XP+SKD,YP,XP+SKD,YP+SKV)
              THEN TEXTKRES:=TRUE;
              IF LINEKRES(XP+SKD,YP+SKV,XP,YP+SKV)
              THEN TEXTKRES:=TRUE;
              IF LINEKRES(XP,YP+SKV,XP,YP) THEN TEX
              TKRES:=TRUE;
              END;
              END;
              PROCEDURE LD1R(HL,DE,BC:INTEGER);
              BEGIN
                IF BC>0 THEN BEGIN
                  POKE (23600,HL);POKE (23602,DE);POKE (2
                  3604,BC);
                  INLINE (*2A,*30,*5C,*ED,*5B,*32,*5C,*ED
                  ,*4B,*34,*5C);
                  INLINE (*ED,*B0);
                  END;
                  END;
                  PROCEDURE LDDR(HL,DE,BC:INTEGER);
                  BEGIN
                    IF BC>0 THEN BEGIN
                      POKE (23600,HL);POKE (23602,DE);POKE (2360
                      4,BC);
                      INLINE (*2A,*30,*5C,*ED,*5B,*32,*5C,*ED
                      ,*4B,*34,*5C);
                      INLINE (*ED,*B0);
                      END;
                      END;
                      FUNCTION POCETPRVKU(BLOK:INTEGER;OBJ:OBJ
                      T):INTEGER;
                      BEGIN
                        POCETPRVKU:=UK[BLOK].POCET[OBJ];
                        END;
                        PROCEDURE POINTERS(BL,M:INTEGER;OBJ:OBJ
                        T);
                        VAR N:INTEGER;
                        POBJ:OBJT;
                        BEGIN
                          IF OBJ<BLOK THEN BEGIN
                            FOR POBJ:=SUCC(OBJ) TO BLOK DO
                              UK[BL].PRVKY[POBJ]:=UK[BL].PRVKY[POBJ
                              ]+M;
                              END;
                              IF BL<PBLMAX THEN BEGIN
                                FOR N:=BL+1 TO PBLMAX DO BEGIN
                                  FOR POBJ:=LINE TO BLOK DO
                                    UK[N].PRVKY[POBJ]:=UK[N].PRVKY[POBJ]
                                    +M;
                                    END;
                                    END;
                                    ENDOF:=ENDOF+M;
                                    END;
                                    END;
                                    PROCEDURE ADLEN(OBJ:OBJT;VAR N,M:INTEGE
                                    R);
                                    BEGIN
                                      CASE OBJ OF
                                        LINE:BEGIN N:=ADDR(PLINE);M:=4;END;
                                        ARC:BEGIN N:=ADDR(PARC);M:=12;END;
```

```

TEXT:BEGIN N:=ADDR(PTEXT);M:=10;END;
BLOK:BEGIN N:=ADDR(PBLOK);M:=5;END;
END;
PROCEDURE NPRVEK(BL,O:INTEGER;OBJ:OBJT)
;
VAR N,M:INTEGER;
BEGIN
ADLEN(OBJ,N,M);
LDIR((O-1)*M+UK(BL).PRVKY[OBJ],N,M);
END;
PROCEDURE VLOZ(BL:INTEGER;OBJ:OBJT);
VAR N,M:INTEGER;
BEGIN
IF ENDOF<TOPMEM THEN BEGIN
ADLEN(OBJ,N,M);
WITH UK(BL) DO BEGIN
LDDR(ENDOF,ENDOF+M,ENDOF-PRVKY[OBJ]-M
*POCET(OBJ)+1);
LDIR(N,PRVKY[OBJ]+M*POCET(OBJ),M);
POCET(OBJ):=POCET(OBJ)+1;
END;
POINTERS(BL,M,OBJ);END
ELSE BEGIN
AT(22,0);WRITE('Memory full !!!
');END;
END;
PROCEDURE SMAZ(BL,M:INTEGER;OBJ:OBJT);
VAR N,O:INTEGER;
BEGIN
ADLEN(OBJ,O,N);
WITH UK(BL) DO
LDIR(PRVKY[OBJ]+M*N,PRVKY[OBJ]+(M-1)*N,
ENDOF-PRVKY[OBJ]-M*N);
N:=N;POINTERS(BL,N,OBJ);
UK(BL).POCET(OBJ):=UK(BL).POCET(OBJ)-1;
END;
PROCEDURE ZMEN(BL,O:INTEGER;OBJ:OBJT);
VAR M,N:INTEGER;
BEGIN
ADLEN(OBJ,N,M);
LDIR(N,UK(BL).PRVKY[OBJ]+M*(O-1),M);
END;
FUNCTION BLOKKRES(BLK:BLOK):BOOLEAN;
VAR X1,Y1,X2,Y2,M,N,A,BL:INTEGER;
BEGIN
BL:=ORD(BLK.CBL);
BLOKKRES:=FALSE;
N:=POCETPRVKU(BL,LINE);M:=1;
WHILE N>0 DO BEGIN
NPRVEK(BL,M,LINE);
IF PLINE.X=MAXINT THEN BEGIN
A:=PLINE.Y;
NPRVEK(BL,M+1,LINE);
X1:=PLINE.X;Y1:=PLINE.Y;
NPRVEK(BL,M+2,LINE);
X2:=PLINE.X;Y2:=PLINE.Y;
M:=M+2;N:=N-2;
END
ELSE BEGIN
X1:=X2;Y1:=Y2;X2:=PLINE.X;Y2:=PLINE.
Y;
END;
IF LINETL(X1+BLK.X,Y1+BLK.Y,X2+BLK.X,Y
2+BLK.Y,A) THEN BEGIN
BLOKKRES:=TRUE;
IF (GBL=BL) AND (KRES=FALSE) THEN BE
GIN
PLINE.X:=PLINE.X+20000;ZMEN(BL,
M,LINE);END;
END;
N:=N-1;M:=M+1;
END;
N:=POCETPRVKU(BL,ARC);X1:=BLK.X;Y1:=BLK
.Y;
IF N>0 THEN BEGIN
FOR M:=1 TO N DO BEGIN
NPRVEK(BL,M,ARC);
WITH PARC DO BEGIN
XC:=XC+X1;YC:=YC+Y1;
XS:=XS+X1;YS:=YS+Y1;
END;
IF ARCKRES(PARC) THEN BEGIN
BLOKKRES:=TRUE;NPRVEK(BL,M,ARC);
IF (GBL=BL) AND (KRES=FALSE) THEN BE
GIN
PARC.XC:=PARC.XC+20000;ZMEN(BL,M,AR
C);END;
END;
END;
END;

```

```

N:=POCETPRVKU(BL,TEXT);
IF N>0 THEN BEGIN
FOR M:=1 TO N DO BEGIN
NPRVEK(BL,M,TEXT);
WITH PTEXT DO BEGIN
XP:=XP+X1;YP:=YP+Y1;END;
IF TEXTKRES(PTEXT) THEN BEGIN
BLOKKRES:=TRUE;NPRVEK(BL,M,TEXT);
IF (GBL=BL) AND (KRES=FALSE) THEN BE
GIN
PTEXT.XP:=PTEXT.XP+20000;ZMEN(BL,M,T
EXT);END;
END;
END;
END;
N:=POCETPRVKU(BL,BLOK);
IF N>0 THEN BEGIN
FOR M:=1 TO N DO BEGIN
NPRVEK(BL,M,BLOK);
WITH PBLOK DO BEGIN
X:=X+X1;Y:=Y+Y1;END;
IF BLOKKRES(PBLOK) THEN BEGIN
BLOKKRES:=TRUE;NPRVEK(BL,M,BLOK);
IF (GBL=BL) AND (KRES=FALSE) THEN BE
GIN
PBLOK.X:=PBLOK.X+20000;ZMEN(BL,M,BL
OK);END;
END;
END;
END;
PROCEDURE KRESLI(BK:BLOK;KR:BOOLEAN);
VAR RESS:BOOLEAN;
BEGIN
KRES:=KR;GBL:=ORD(BK.CBL);
RESS:=BLOKKRES(BK);
KRES:=TRUE;
END;
PROCEDURE PAUSE(N:INTEGER);
VAR M:INTEGER;
BEGIN
FOR M:=1 TO N DO
END;
PROCEDURE DRAW(X,Y,X1,Y1:INTEGER;O:INTE
GER);
BEGIN
DRFT;PLOT(X,Y,O);PLOT(X1,Y1,O);DOT;
END;
PROCEDURE KURZOR(X,Y:INTEGER);
BEGIN
DRAW(X-2,Y,X+2,Y,9);DRAW(X,Y-2,X,Y+2,9
);
END;
FUNCTION POHYBKUR(XR,YR:INTEGER;KR:BOOL
EAN):CHAR;
VAR A:CHAR;
TIME,R,N:INTEGER;
PROCEDURE KRIZW;
BEGIN
KURZOR(XCUR,YCUR);
DRAW(XCUR,1,XCUR,175,9);DRAW(1,YCUR,20
7,YCUR,9);
IF KR THEN BEGIN
DRAW(XR,YR,XR,YCUR,9);DRAW
(XR,YR,XCUR,YR,9);END;
END;
BEGIN
R:=0;TIME:=0;
REPEAT
A:=INCH;
IF A=CHR(0) THEN BEGIN
REPEAT
A:=INCH;TIME:=TIME+1;
IF TIME>3500 THEN BEGIN
KRIZW;TIME:=0;
REPEAT
A:=INCH;
UNTIL A<>CHR(0);
KRIZW;END;
UNTIL A<>CHR(0);
TIME:=0;R:=0;END;
R:=R+RYCHLOSTI[R DIV 20+1].P;
IF A='O' THEN XCUR:=XCUR-RYCHLOSTI[R D
IV 20+1].XRY;
IF A='P' THEN XCUR:=XCUR+RYCHLOSTI[R D
IV 20+1].XRY;
IF A='Q' THEN YCUR:=YCUR+RYCHLOSTI[R D
IV 20+1].YRY;
IF A='A' THEN YCUR:=YCUR-RYCHLOSTI[R D
IV 20+1].YRY;
IF (XCUR<2) OR (XCUR>205) OR (YCUR<2)

```

```

OR (YCUR>173) THEN BEGIN
XCUR:=XCUR;YCUR:=YCUR;END;
WRITE(CHR(21),CHR(0));AT(0,0);WRITE(((X
CUR-PANX*SCALE)/SCALE*0.125):8:3);
AT(0,10);WRITE(((YCUR-PANY*SCALE)/SCALE
*0.125):8:3);
KURZOR(XCUR,YCUR);KURZOR(XCUR,YCUR);
XCUR:=XCUR;YCUR:=YCUR;
PAUSE(900);
UNTIL NOT((A='O') OR (A='P') OR (A='Q')
OR (A='A'));
POHYBKUR:=A;WRITE(CHR(21),CHR(0));
REPEAT UNTIL INCH=CHR(0);
END;
PROCEDURE ZOBRAZCAST(C:INTEGER);FORWARD
;
PROCEDURE OBNOVNAPIS;FORWARD;
PROCEDURE ZOBRAZCAST;
VAR N:INTEGER;
BEGIN
WRITE(CHR(21),CHR(0));
FOR N:=C TO C+16 DO BEGIN
AT(N+C+2,26);
IF N>PBLAKT+POCPRIK THEN WRITE('
')
ELSE WRITE(JMENAIN
));
END;
END;
END;
PROCEDURE INVERSE(C:INTEGER;SW:BOOLEAN)
;
BEGIN
AT(C,26);
IF SW THEN WRITE(CHR(17),CHR(2))
ELSE WRITE(CHR(17),CHR(0));
WRITE(CHR(21),CHR(1),' ');
WRITE(CHR(21),CHR(0),CHR(17),CHR(0));
END;
FUNCTION FCEMNU(B:INTEGER):CHAR;
VAR A:CHAR;
BEGIN
IF MNUA<B THEN BEGIN
MNUA:=B;ZOBRAZCAST(MNUA);END;
INVERSE(MNUR+2,TRUE);
REPEAT UNTIL INCH=CHR(0);
REPEAT
A:=INCH;
UNTIL A<>CHR(0);
IF A='O' THEN BEGIN
MNUA:=MNUA-10;
IF MNUA<B THEN MNUA:=B;
ZOBRAZCAST(MNUA);INVERSE(MNUR+2,TRUE)
;END;
IF A='P' THEN BEGIN
MNUA:=MNUA+10;
IF MNUA>(PBLAKT+POCPRIK-16) THEN MNUA
:=PBLAKT+POCPRIK-16;
ZOBRAZCAST(MNUA);INVERSE(MNUR+2,TRUE)
;END;
IF A='Q' THEN BEGIN
MNUR:=MNUR-1;
IF MNUR<0 THEN MNUR:=0;
INVERSE(MNUR+3,FALSE);INVERSE(MNUR+2,
TRUE);END;
IF A='A' THEN BEGIN
MNUR:=MNUR+1;
IF MNUR>16 THEN MNUR:=16;
INVERSE(MNUR+1,FALSE);INVERSE(MNUR+2,
TRUE);END;
PAUSE(900);
UNTIL NOT((A='O') OR (A='P') OR (A='Q')
OR (A='A'));
FCEMNU:=A;INVERSE(MNUR+2,FALSE);
REPEAT UNTIL INCH=CHR(0);
IF (MNUA+MNUR<B) OR (MNUA+MNUR>POCPRIK+
PBLAKT) THEN BEGIN
MNUA:=B;MNUR:=0;ZOBRAZCAST(B);END;
END;
PROCEDURE OBNOVNAPIS;
BEGIN
AT(22,0);WRITE(JMENADP[1]),' MENU
',JMENADP[2]),' ',JMENADP[3]);
END;
FUNCTION CPRIK(A:CHAR):INTEGER;
BEGIN
CPRIK:=0;
IF A='X' THEN BEGIN
AT(22,0);WRITE('SELECT CREATE DEFINE S
ELECT');
A:=FCEMNU(1);

```

```

IF (A='Z') OR (A='M') THEN CPRIK:=MNUA+MNUR;
IF A='N' THEN BEGIN
  AT(22,0);WRITE('Select key (Z,N,M)');
  REPEAT UNTIL INCH<CHR(0);
  A:=INCH;
  IF A='Z' THEN DP[1]:=MNUA+MNUR;
  IF A='N' THEN DP[2]:=MNUA+MNUR;
  IF A='M' THEN DP[3]:=MNUA+MNUR;
  REPEAT UNTIL INCH<CHR(0);
  END;
OBNOVNAPIS;END
ELSE BEGIN
  IF A='Z' THEN CPRIK:=DP[1];
  IF A='N' THEN CPRIK:=DP[2];
  IF A='M' THEN CPRIK:=DP[3];
  END;
END;
PROCEDURE INITIAL;
PROCEDURE INITDISP;
BEGIN
  PANX:=500;PANY:=400;SCALE:=1/5;KRES:=T
  RUE;
  X3:=0;Y3:=0;X4:=100;Y4:=100;
  END;
PROCEDURE INITDM;
VAR N:INTEGER;POBJ:OBJT;
BEGIN
  FOR N:=1 TO PBLMAX DO BEGIN
    FOR POBJ:=LINE TO BLOK DO BEGIN
      UK[N].PRVKY[POBJ]:=OFFSET;
      UK[N].POCET[POBJ]:=0;
    END;
  END;
  ENDOF:=OFFSET;
  END;
PROCEDURE INITMNU;
BEGIN
  XCUR:=100;XCURS:=100;YCUR:=80;YCURS:=80
  RYCHLOST[1].XRY:=1;RYCHLOST[1].YRY:=1
  RYCHLOST[1].P:=4;
  RYCHLOST[2].XRY:=2;RYCHLOST[2].YRY:=2
  RYCHLOST[2].P:=2;
  RYCHLOST[3].XRY:=7;RYCHLOST[3].YRY:=7
  RYCHLOST[3].P:=0;
  PAGE;KURZOR(XCUR,YCUR);
  PBLAKT:=0;CURBLOK:=1;CURLT:=101;
  FOR GBL:=1 TO JMCELKEM DO JMENA[GBL]:=
  JMENA[1]:='OPEN';JMENA[2]:='CURR';J
  MENA[5]:='MOVE';
  JMENA[6]:='COPY';JMENA[7]:='ERASE';J
  MENA[8]:='L.TYPE';
  JMENA[11]:='ZOOM';JMENA[12]:='PAN'
  JMENA[13]:='CURSOR';JMENA[15]:='SAVE'
  JMENA[16]:='LOAD';JMENA[19]:='LINE'
  JMENA[20]:='ARC';JMENA[21]:='TEXT'
  JMENA[24]:='blok 1';
  JMENA[18]:='de18';JMENA[22]:='de22'
  POKE(23659,CHR(1));
  MNUA:=1;MNUR:=15;ZOBRAZCAST(1);
  DP[1]:=7;DP[2]:=21;DP[3]:=19;
  OBNOVNAPIS;
  END;
BEGIN
  INITDISP;
  INITDM;
  INITMNU;
  END;
PROCEDURE KUR;
BEGIN KURZOR(XCUR,YCUR);WRITE(CHR(21),C
  HR(0));END;
FUNCTION XSK:INTEGER;
BEGIN
  XSK:=ROUND((XCUR-PANX*SCALE)/SCALE);END
FUNCTION YSK:INTEGER;
BEGIN
  YSK:=ROUND((YCUR-PANY*SCALE)/SCALE);END
PROCEDURE PRLINE;
VAR A:CHAR;
  XM,YM:INTEGER;
  B:BOOLEAN;

```

```

BEGIN
  PLINE.X:=MAXINT;PLINE.Y:=CURLT;VLOZ(CUR
  BLOK,LINE);
  PLINE.X:=XSK;PLINE.Y:=YSK;XM:=XSK;YM:=Y
  SK;VLOZ(CURBLOK,LINE);
  AT(22,0);WRITE('To point
  ');
  A:=POHYBKUR(0,0,FALSE);
  PLINE.X:=XSK;PLINE.Y:=YSK;
  KUR;
  B:=LINETL(XM,YM,XSK,YSK,CURLT);VLOZ(CUR
  BLOK,LINE);
  KUR;
  XM:=XSK;YM:=YSK;
  AT(22,0);WRITE('END Next point ');
  WHILE NOT(POHYBKUR(0,0,FALSE)='Z') DO B
  EGIN
    PLINE.X:=XSK;PLINE.Y:=YSK;VLOZ(CURBLOK
    ,LINE);
    KUR;
    B:=LINETL(XM,YM,XSK,YSK,CURLT);XM:=XSK
    ;YM:=YSK;
    KUR;
    END;
  OBNOVNAPIS;
  END;
PROCEDURE PRARC;
VAR F:CHAR;B:BOOLEAN;
BEGIN
  AT(22,0);WRITE('Center point
  ');
  F:=POHYBKUR(0,0,FALSE);
  PARC.XC:=XSK;PARC.YC:=YSK;
  AT(22,0);WRITE('Start point ');
  F:=POHYBKUR(0,0,FALSE);
  PARC.XS:=XSK;PARC.YS:=YSK;
  AT(22,0);WRITE('Enter angle ');
  AT(22,13);READLN;READ(PARC.U);
  PARC.U:=ROUND(PARC.U/180*3.1415*1000);
  PARC.A:=CURLT;
  KUR;
  VLOZ(CURBLOK,ARC);B:=ARCKRES(PARC);OBNO
  VNAPIS;
  KUR;
  END;
PROCEDURE PRTEXT;
VAR B:BOOLEAN;
  C:REAL;
  BEGIN
    AT(22,0);WRITE('Enter text
    ');
    AT(22,12);READLN;READ(PTEXT.T);
    AT(22,0);WRITE('Size
    mm
    ');
    AT(22,5);READLN;READ(C);
    PTEXT.VELKOST:=CHR(ROUND(ABS(C)*2));
    PTEXT.XP:=XSK;PTEXT.YP:=YSK;VLOZ(CURBLO
    K,TEXT);
    KUR;
    B:=TEXTKRES(PTEXT);
    KUR;
    OBNOVNAPIS;
    END;
  PROCEDURE PRBLOK(A:INTEGER);
  VAR B:BOOLEAN;
  BEGIN
    IF (A<POCPRIK) AND (A<POCPRIK+PBLAKT)
    AND (A<CURBLOK+POCPRIK) THEN BEGIN
      PBLOK.X:=XSK;PBLOK.Y:=YSK;PBLOK.CBL:=C
      HR(A-POCPRIK);
      KUR;
      VLOZ(CURBLOK,BLOK);B:=BLOKKRES(PBLOK);
      KUR;
      END;
    END;
  PROCEDURE CLWA;
  VAR N:INTEGER;
  BEGIN
    WRITE(CHR(21),CHR(0));
    FOR N:=0 TO 21 DO BEGIN
      AT(N,0);WRITE('
      ');
    END;
  END;
  PROCEDURE REDRAW;
  BEGIN
    PBLOK.X:=0;PBLOK.Y:=0;PBLOK.CBL:=CHR(C
    URLBLOK);
    KRESLI(PBLOK,TRUE);
    KUR;
    END;

```

```

PROCEDURE PRCURR;
VAR A:CHAR;
  N,M:INTEGER;
  BEGIN
    AT(22,0);WRITE('Backgr. Current block
    ');
    CLWA;
    REPEAT
      A:=FCENMU(POCPRIK+1);
      IF A='Z' THEN BEGIN
        CURBLOK:=MNUA+MNUR-POCPRIK;
        REDRAW;
        FOR N:=0 TO 103 DO BEGIN
          FOR M:=0 TO 87 DO BEGIN
            PLOT(N*2,M*2,0);PLOT(N*2,M*2,9)
            ;
            PLOT(N*2+1,M*2+1,0);PLOT(N*2+1,
            M*2+1,9);
            END;
          END;
        END;
      ELSE BEGIN
        CURBLOK:=MNUA+MNUR-POCPRIK;REDRAW
        ;END;
      UNTIL NOT(A='Z');
      OBNOVNAPIS;
      END;
    PROCEDURE PROPEN;
    BEGIN
      IF PBLAKT=PBLMAX THEN BEGIN
        AT(22,0);WRITE('Too many blocks
        ');
        PAUSE(30000);END
      ELSE BEGIN
        PBLAKT:=PBLAKT+1;CURBLOK:=PBLAKT;
        AT(22,0);WRITE('Name
        ');
        AT(22,6);READLN;READ(JMENA(POCPRI
        K+CURBLOK));
        ZOBRAZCAST(PBLAKT+POCPRIK-15);CLW
        A;
        END;
      OBNOVNAPIS;KUR;
      END;
    PROCEDURE WINDOW;FORWARD;
    PROCEDURE ZJISTILI(BL,P:INTEGER;VAR N,M
    :INTEGER);
    VAR O:INTEGER;
    BEGIN
      N:=P;
      REPEAT
        N:=N-1;NPRVEK(BL,N,LINE);
        UNTIL PLINE.X=MAXINT;
        M:=P;O:=POCETPRVKU(BL,LINE);
        REPEAT
          M:=M+1;NPRVEK(BL,M,LINE);
          UNTIL (PLINE.X=MAXINT) OR (M=O+1);
          M:=M-1;
        END;
        PROCEDURE ERASE(BL:INTEGER);
        VAR M,N,O:INTEGER;
        OBJ:OBJT;
        BEGIN
          WINDOW;
          M:=1;N:=POCETPRVKU(BL,LINE);
          WHILE N>0 DO BEGIN
            NPRVEK(BL,M,LINE);
            IF (PLINE.X>10000) AND (PLINE.X<MAXINT
            ) THEN BEGIN
              ZJISTILI(BL,M,N,O);
              FOR M:=N TO 0 DO SMAZ(BL,N,LINE);
              N:=POCETPRVKU(BL,LINE)+1;M:=0;
              END;
            N:=N-1;M:=M+1;
            END;
          FOR OBJ:=ARC TO BLOK DO BEGIN
            M:=1;
            FOR N:=1 TO POCETPRVKU(BL,OBJ) DO BEGI
            N
              NPRVEK(BL,M,OBJ);
              CASE OBJ OF
                ARC:O:=PARC.XC;
                TEXT:O:=PTEXT.XP;
                BLOK:O:=PBLOK.X
              END;
              IF O>10000 THEN BEGIN
                SMAZ(BL,M,OBJ);M:=M-1;END;
                M:=M+1;
              END;
            END;
            CLWA;REDRAW;OBNOVNAPIS;
            END;

```



```

PROCEDURE WINDOW;
VAR A:CHAR;
    XR,YR:INTEGER;
BEGIN
    AT(22,0);WRITE('Left down corner
    ');
    A:=POHYBKUR(0,0,FALSE);
    X3:=XCUR;Y3:=YCUR;XR:=XCUR;YR:=YCUR;
    AT(22,0);WRITE('Right up corner
    ');
    A:=POHYBKUR(XR,YR,TRUE);
    X4:=XCUR;Y4:=YCUR;
    PBLOK.X:=0;PBLOK.Y:=0;PBLOK.CBL:=CHR(CU
    RBLOK);
    KRESLI(PBLOK,FALSE);
END;
PROCEDURE PMZAC;
VAR A:CHAR;
    XR,YR:INTEGER;
BEGIN
    WINDOW;
    AT(22,0);WRITE('New left down corner
    ');
    A:=POHYBKUR(0,0,FALSE);
    X4:=ROUND((XCUR-PANX*SCALE)/SCALE)-ROUN
    D((X3-PANX*SCALE)/SCALE);
    Y4:=ROUND((YCUR-PANY*SCALE)/SCALE)-ROUN
    D((Y3-PANY*SCALE)/SCALE);
END;
PROCEDURE MOVECOPY(MOVE:BOOLEAN;BL,DX,D
    Y:INTEGER);
VAR M,N,O,P,Q:INTEGER;
    OBJ:OBJT;
BEGIN
    Q:=POCETPRVKU(BL,LINE);
    FOR N:=1 TO Q DO BEGIN
        NPRVEK(BL,N,LINE);
        IF (PLINE.X>10000) AND (PLINE.X<MAXINT
        ) THEN BEGIN
            ZJSTILI(BL,N,M,O);
            IF MOVE=FALSE THEN BEGIN NPRVEK(BL,M,
            LINE);VLOZ(BL,LINE);END;
            FOR P:=M+1 TO O DO BEGIN
                NPRVEK(BL,P,LINE);
                IF PLINE.X>10000 THEN PLINE.X:=PLINE
                .X-20000;
                ZMEN(BL,P,LINE);
                PLINE.X:=PLINE.X+DX;PLINE.Y:=PLINE.Y
                +DY;
                IF MOVE THEN ZMEN(BL,P,LINE)
                ELSE VLOZ(BL,LINE);
            END;
        END;
    END;
    FOR OBJ:=ARC TO BLOK DO BEGIN
        Q:=POCETPRVKU(BL,OBJ);
        FOR N:=1 TO Q DO BEGIN
            NPRVEK(BL,N,OBJ);
            CASE OBJ OF
                ARC:0:=PARC.XC;
                TEXT:0:=PTEXT.XP;
                BLOK:0:=PBLOK.X
            END;
            IF O>10000 THEN BEGIN
                CASE OBJ OF
                    ARC:BEGIN
                        PARC.XC:=PARC.XC-20000;ZMEN(B
                        L,N,OBJ);
                        WITH PARC DO BEGIN
                            XC:=XC+DX;YC:=YC+DY;
                            XS:=XS+DX;YS:=YS+DY;
                        END;
                    END;
                    TEXT:BEGIN
                        PTEXT.XP:=PTEXT.XP-20000;ZMEN
                        (BL,N,OBJ);
                        PTEXT.XP:=PTEXT.XP+DX;PTEXT.Y
                        P:=PTEXT.YP+DY;
                    END;
                    BLOK:BEGIN
                        PBLOK.X:=PBLOK.X-20000;ZMEN(B
                        L,N,OBJ);
                        PBLOK.X:=PBLOK.X+DX;PBLOK.Y:=
                        PBLOK.Y+DY;
                    END
                END;
                IF MOVE THEN ZMEN(BL,N,OBJ)
                ELSE VLOZ(BL,OBJ);
            END;
        END;
    END;
    CLWA;REDRAW;OBNOVNAPIS;
END;

```

```

PROCEDURE VYPAN(ZPX,ZPY:REAL);FORWARD;
PROCEDURE PRLINETYPE;
VAR A:INTEGER;
BEGIN
    AT(22,0);WRITE('Number of parcel lines
    ');
    AT(22,24);READLN;READ(A);
    IF (A>99) OR (A<1) THEN A:=1;
    CURLT:=A;
    AT(22,0);WRITE('Distance between lines
    ');
    AT(22,24);READLN;READ(A);
    IF (A>99) OR (A<0) THEN A:=1;
    CURLT:=CURLT+A*100;
    OBNOVNAPIS;
END;
PROCEDURE PRPAN;
VAR ZPX,ZPY:REAL;
BEGIN
    AT(22,0);WRITE('Enter X coord.
    ');
    AT(22,17);READLN;READ(ZPX);
    AT(22,0);WRITE('Enter Y coord.
    ');
    AT(22,17);READLN;READ(ZPY);
    ZPX:=ZPX;ZPY:=ZPY;
    VYPAN(ZPX,ZPY);
    CLWA;REDRAW;OBNOVNAPIS;
END;
PROCEDURE PRZOOM;
VAR SC,ZX,ZY:REAL;
BEGIN
    REPEAT
        AT(22,0);WRITE('Enter scale 1:
        ');
        AT(22,14);READLN;READ(SC);
        UNTIL (SC>=1) AND (SC<100);
        SC:=1/SC;
        ZX:=(PANX-100/SCALE)/8;
        ZY:=(PANY-80/SCALE)/8;
        SCALE:=SC;
        VYPAN(ZX,ZY);
        CLWA;REDRAW;OBNOVNAPIS;
    END;
    PROCEDURE VYPAN;
    BEGIN
        PANX:=ROUND(ZPX*8+100/SCALE);
        PANY:=ROUND(ZPY*8+80/SCALE);
    END;
    PROCEDURE PRSAVE;
    VAR A:CHAR;
        OD,DOB:INTEGER;
        NAME:ARRAY[1..8] OF CHAR;
    BEGIN
        NAME:='';
        AT(22,0);WRITE('First block
        ');
        A:=FCENIU(POCPRIK+1);
        OD:=MNUA+MNUR;
        AT(22,0);WRITE('Last');
        A:=FCENIU(OD);
        DOB:=MNUA+MNUR;
        AT(22,0);WRITE('Name
        ');
        AT(22,5);READLN;READ(NAME);
        NAME[6]:='.';NAME[7]:='J';NAME[8]:='M';
        TOUT(NAME,ADDR(JMENA)+6*(OD-1),6*(DOB-O
        D+1));
        OD:=OD-POCPRIK;DOB:=DOB-POCPRIK;
        NAME[7]:='U';NAME[8]:='K';
        TOUT(NAME,ADDR(UK)+16*(OD-1),16*(DOB-OD
        +1));
        NAME[7]:='V';NAME[8]:='S';
        TOUT(NAME,UK[OD].PRVKY[LINE],UK[DOB+1].
        PRVKY[LINE]-UK[OD].PRVKY[LINE]);
        PAGE;
        ZOBRAZCAST(MNUA);OBNOVNAPIS;
        PBLOK.X:=0;PBLOK.Y:=0;PBLOK.CBL:=CHR(CU
        RBLOK);
        KRESLI(PBLOK,TRUE);KURZOR(XCUR,YCUR);
    END;
    PROCEDURE PRLOAD;
    VAR N,M,DIF:INTEGER;
        NAME:ARRAY[1..8] OF CHAR;
        POBJ:OBJT;
    BEGIN
        NAME:='';
        AT(22,0);WRITE('Name
        ');
        AT(22,5);READLN;READ(NAME);
        NAME[6]:='.';NAME[7]:='J';NAME[8]:='M';
        AT(22,0);

```

```

    TIN(NAME,ADDR(JMENA)+6*(POCPRIK+PBLAKT)
    );
    N:=PBLAKT+POCPRIK+1;
    WHILE (N<=JMCELKEM) AND (JMENA[N]<>
    ') DO N:=N+1;
    N:=N-1-POCPRIK;
    NAME[7]:='U';NAME[8]:='K';
    TIN(NAME,ADDR(UK)+16*PBLAKT);
    DIF:=UK[PBLAKT+1].PRVKY[LINE]-ENDOF;
    IF UK[N].PRVKY[BLOK]+5*UK[N].POCET(BLOK
    )-DIF<TOPMEM THEN BEGIN
        FOR M:=PBLAKT+1 TO N DO BEGIN
            FOR POBJ:=LINE TO BLOK DO
                UK[M].PRVKY[POBJ]:=UK[M].PRVKY[POBJ]-
                DIF;
            END;
            DIF:=UK[N].PRVKY[BLOK]+5*UK[N].POCET(BL
            OK);
            IF N<PBLMAX THEN BEGIN
                FOR M:=N+1 TO PBLMAX DO BEGIN
                    FOR POBJ:=LINE TO BLOK DO UK[M].PRVKY
                    [POBJ]:=DIF;
                END;
            END;
            NAME[7]:='V';NAME[8]:='S';
            TIN(NAME,ENDOF);
            PBLAKT:=N;ENDOF:=DIF;
        END
    ELSE BEGIN
        AT(22,0);WRITE('No memory for file
        ');
        PAUSE(30000);END;
    PAGE;
    ZOBRAZCAST(MNUA);OBNOVNAPIS;
    PBLOK.X:=0;PBLOK.Y:=0;PBLOK.CBL:=CHR(CU
    RBLOK);
    KRESLI(PBLOK,TRUE);KURZOR(XCUR,YCUR);
    END;
    BEGIN
        INITIAL;
        REPEAT
            CISLO:=CPRIK(POHYBKUR(0,0,FALSE));
            IF CISLO=19 THEN PRLINE;
            IF CISLO=20 THEN PRARC;
            IF CISLO=21 THEN PRTEXT;
            IF CISLO>POCPRIK THEN PRBLOK(CISLO);
            IF CISLO=1 THEN PROPEN;
            IF CISLO=2 THEN PRCURR;
            IF CISLO=7 THEN ERASE(CURBLOK);
            IF CISLO=5 THEN BEGIN
                PMZAC;
                MOVECOPY(TRUE,CURBLOK
                ,X4,Y4);END;
            IF CISLO=6 THEN BEGIN
                PMZAC;
                MOVECOPY(FALSE,CURBLO
                K,X4,Y4);END;
            IF CISLO=8 THEN PRLINETYPE;
            IF CISLO=11 THEN PRZOOM;
            IF CISLO=12 THEN PRPAN;
            IF CISLO=15 THEN PRSAVE;
            IF CISLO=16 THEN PRLOAD;
            UNTIL FALSE;
        END.

```

## Výpis 2. Program ZCF (929-V2)

```

PROGRAM A;
CONST PBLMAX=60;
    POCPRIK=23;
    JMCELKEM=83;
    OFFSET:=C350;TOPMEM:=EA00;
TYPE LINET=RECORD
    X,Y:INTEGER;END;
    ARCT=RECORD XC,YC,XS,YS,A,U:INTEGE
    R;END;
    TEXTT=RECORD VELIKOST:CHAR;XP,YP:I
    NTEGER;T:ARRAY[1..5] OF CHAR;END;
    OBJT=(LINE,ARC,TEXT,BLOK);
    UKAZATELE=ARRAY[1..PBLMAX] OF RE
    CORD
    P
    RVKY=ARRAY[LINE..BLOK] OF INTEGER;
    P
    OCET=ARRAY[LINE..BLOK] OF INTEGER;
    E
    ND;
    BLOKT=RECORD X,Y:INTEGER;CBL:CHAR;
    END;

```



```

NAMET-ARRAY[1..10] OF CHAR;
STRING-ARRAY[1..10] OF CHAR;
MALYSTRING-ARRAY[1..5] OF CHAR;
VAR PLINE:LINET;PARC:ARCT;PTEXT:TEXTT;P
BLOK:BLOKT;
PBLAKT:INTEGER;
UK:UKAZATELE;ENDOF:INTEGER;
JMEANA:ARRAY[1..JMCELKEM,1..6] OF CH
AR;
CISRET:STRING;
PROCEDURE CISPREV(C:REAL);
VAR A,B:REAL;N:INTEGER;
FUNCTION POMC(B:REAL):CHAR;
BEGIN
POMC:=CHR(ROUND((A-(ENTIER(A/B/10)*10+B
+FRAC(A/B)*B)/B)+48);
END;
BEGIN
A:=ABS(C);B:=1000;CISRET:='';
IF C<0 THEN CISRET[1]:='-';
FOR N:=3 DOWNT0 DO BEGIN CISRET[5-N]:
=POMC(B);B:=B/10;END;
CISRET[6]:='-';
FOR N:=1 DOWNT-2 DO BEGIN CISRET[6-N
]:=POMC(B);B:=B/10;END;
END;
PROCEDURE ODVYS4(T:STRING);
VAR N:INTEGER;
BEGIN
FOR N:=1 TO 10 DO BEGIN
POKE(16384,T[N]);
INLINE(*3A,0,*40,*CD,*SD,*EC);END;
END;
PROCEDURE ODVYS5(T:STRING);
VAR N:INTEGER;
BEGIN
T[10]:=CHR(13);
FOR N:=1 TO 10 DO BEGIN
POKE(16384,T[N]);
INLINE(*3A,0,*40,*CD,*60,*EC);END;
END;
PROCEDURE ODVENT;
VAR S:STRING;
BEGIN
S:='';
S[10]:=CHR(13);ODVYS4(S);
END;
PROCEDURE ODVCAR;
BEGIN
ODVYS4('');
END;
PROCEDURE MA(X,Y:REAL);
BEGIN
ODVYS4('MA');
CISPREV(X);ODVYS4(CISRET);
ODVCAR;
CISPREV(Y);ODVYS4(CISRET);
ODVENT;
END;
PROCEDURE VA(X,Y:REAL);
BEGIN
ODVYS4('VA');
CISPREV(X);ODVYS4(CISRET);
ODVCAR;
CISPREV(Y);ODVYS4(CISRET);
ODVENT;
END;
PROCEDURE OG(X,Y:REAL);
BEGIN
ODVYS4('OG');
CISPREV(X);ODVYS4(CISRET);ODVCAR;CISPRE
V(Y);ODVYS4(CISRET);ODVENT;
END;
PROCEDURE MF;
BEGIN ODVYS4('MF');ODVENT;END;
PROCEDURE SC(S:REAL);
BEGIN
ODVYS4('SC');CISPREV(S);ODVYS4(
CISRET);ODVENT;END;
PROCEDURE LT(L:INTEGER);
BEGIN ODVYS4('LT');CISPREV(L);O
DVYS4(CISRET);END;
PROCEDURE CS(X,Y:REAL);
BEGIN
ODVYS4('CS');
CISPREV(X);ODVYS4(CISRET);ODVCAR;CISPRE
V(Y);ODVYS4(CISRET);
ODVYS4('');ODVENT;
END;
PROCEDURE AC(A,B,C:REAL);
BEGIN
ODVYS4('AC');

```

```

CISPREV(A);ODVYS4(CISRET);ODVCAR;
CISPREV(B);ODVYS4(CISRET);ODVCAR;
CISPREV(C);ODVYS4(CISRET);ODVENT;
END;
PROCEDURE SPEED(S:INTEGER);
BEGIN
POKE(*EBAD,S);END;
PROCEDURE LINETL(X1,Y1,X2,Y2,A:INTEGER)
;
VAR N,M,MFX,MFY:INTEGER;M1,M2,XR,YR:REA
L;
BEGIN
LT(0);
IF A=2 THEN BEGIN A:=1;LT(3);END;
IF A=3 THEN BEGIN A:=1;LT(2);END;
N:=A MOD 100; A:=A DIV 100;
M1:=SQRT(SQR(X2-X1)+SQR(Y2-Y1));
IF M1=0 THEN M1:=1;
M2:=(Y2-Y1)/M1*A*(N-1)/2;
M1:=(X2-X1)/M1*A*(N-1)/2;
X2:=X2-X1;XR:=X1-M2;
Y2:=Y2-Y1;YR:=Y1-M1;
IF N>1 THEN BEGIN M1:=M1/((N-1)/2);M2:=
M2/((N-1)/2);END;
FOR M:=1 TO N DO BEGIN
X1:=ROUND(XR);Y1:=ROUND(YR);
MA(X1,Y1);
VA(X1+X2,Y1+Y2);
XR:=XR+M2;YR:=YR-M1;
END;
END;
PROCEDURE ARCTL(XC,YC,XS,YS,U,A:INTEGER
);
VAR POCUHEL,KONUHEL,FR,PS:REAL;
M,CAR,VZD:INTEGER;
BEGIN
LT(0);
IF A=2 THEN BEGIN A:=1;LT(3);END;
IF A=3 THEN BEGIN A:=1;LT(2);END;
CAR:=A MOD 100;VZD:=A DIV 100;
PR:=SQRT(SQR(XS-XC)+SQR(YS-YC));
IF U<0 THEN PR:=-PR;
POCUHEL:=ARCTAN((YS-YC)/(XS-XC+0.01));
IF (XS-XC)<0 THEN POCUHEL:=POCUHEL+3.14
159;
KONUHEL:=POCUHEL+U/1000;
IF KONUHEL>6.2832 THEN KONUHEL:=KONUHEL
-6.2832;
IF ABS(U)>6196 THEN KONUHEL:=POCUHEL;
PS:=PR-(CAR-1)/2*VZD;
FOR M:=1 TO CAR DO BEGIN
MA(XC,YC);
AC(PS,POCUHEL,KONUHEL);
PS:=PS+VZD;
END;
END;
PROCEDURE TEXTKRES(VELIKOST:CHAR;XP,YP:
INTEGER;T:MALYSTRING);
VAR N:INTEGER;
VEL:REAL;
T1:ARRAY[1..10] OF CHAR;
BEGIN
VEL:=ORD(VELIKOST)/2*8;
T1:='';
MA(XP,YP);
CS(VEL,VEL);
FOR N:=1 TO 5 DO T1[N]:=T[N];
T1[6]:=CHR(13);
ODVYS(T1);
END;
PROCEDURE LD1R(HL,DE,BC:INTEGER);
BEGIN
IF BC>0 THEN BEGIN
POKE(23600,HL);POKE(23602,DE);POKE(2
3604,BC);
INLINE(*2A,*30,*5C,*ED,*5B,*32,*5C,*ED
,*4B,*34,*5C);
INLINE(*ED,*B0);
END;
END;
PROCEDURE LDDR(HL,DE,BC:INTEGER);
BEGIN
IF BC>0 THEN BEGIN
POKE(23600,HL);POKE(23602,DE);POKE(2360
4,BC);
INLINE(*2A,*30,*5C,*ED,*5B,*32,*5C,*ED
,*4B,*34,*5C);
INLINE(*ED,*B0);
END;
END;
FUNCTION POCETPRVKU(BLOK:INTEGER;OBJ:OB
JT):INTEGER;

```

```

BEGIN
POCETPRVKU:=UK(BLOK).POCET(OBJ);
END;
PROCEDURE POINTERS(BL,M:INTEGER;OBJ:OBJ
T);
VAR N:INTEGER;
POBJ:OBJT;
BEGIN
IF OBJ<BLOK THEN BEGIN
FOR POBJ:=SUCC(OBJ) TO BLOK DO
UK(BL).PRVKY[POBJ]:=UK(BL).PRVKY[POBJ
1+M];
END;
IF BL<PBLMAX THEN BEGIN
FOR N:=BL+1 TO PBLMAX DO BEGIN
FOR POBJ:=LINE TO BLOK DO
UK(N).PRVKY[POBJ]:=UK(N).PRVKY[POBJ]
+M;
END;
END;
ENDOF:=ENDOF+M;
END;
PROCEDURE ADLEN(OBJ:OBJT;VAR N,M:INTEGE
R);
BEGIN
CASE OBJ OF
LINE:BEGIN N:=ADDR(PLINE);M:=4;END;
ARC:BEGIN N:=ADDR(PARC);M:=12;END;
TEXT:BEGIN N:=ADDR(PTEXT);M:=10;END;
BLOK:BEGIN N:=ADDR(PBLOK);M:=5;END
END;
END;
PROCEDURE NPRVEK(BL,O:INTEGER;OBJ:OBJT)
;
VAR N,M:INTEGER;
BEGIN
ADLEN(OBJ,N,M);
LD1R((O-1)*M+UK(BL).PRVKY[OBJ],N,M);
END;
PROCEDURE BLOKKRES(BLK:BLOKT);
VAR X1,Y1,X2,Y2,M,N,A1P,BL:INTEGER;
BEGIN
BL:=ORD(BLK.CBL);
N:=POCETPRVKU(BL,LINE);M:=1;
WHILE N>0 DO BEGIN
NPRVEK(BL,M,LINE);
IF PLINE.X=MAXINT THEN BEGIN
A1P:=PLINE.Y;
NPRVEK(BL,M+1,LINE);
X1:=PLINE.X;Y1:=PLINE.Y;
NPRVEK(BL,M+2,LINE);
X2:=PLINE.X;Y2:=PLINE.Y;
M:=M+2;N:=N-2;
END
ELSE BEGIN
X1:=X2;Y1:=Y2;X2:=PLINE.X;Y2:=PLINE.
Y;
END;
END;
LINETL(X1+BLK.X,Y1+BLK.Y,X2+BLK.X,Y
2+BLK.Y,A1P);
N:=N-1;M:=M+1;
END;
N:=POCETPRVKU(BL,ARC);X1:=BLK.X;Y1:=BLK
.Y;
IF N>0 THEN BEGIN
FOR M:=1 TO N DO BEGIN
NPRVEK(BL,M,ARC);
WITH PARC DO BEGIN
XC:=XC+X1;YC:=YC+Y1;
XS:=XS+X1;YS:=YS+Y1;
ARCTL(XC,YC,XS,YS,U,A);
END;
END;
END;
N:=POCETPRVKU(BL,TEXT);
IF N>0 THEN BEGIN
FOR M:=1 TO N DO BEGIN
NPRVEK(BL,M,TEXT);
WITH PTEXT DO BEGIN
XP:=XP+X1;YP:=YP+Y1;
TEXTKRES(VELIKOST,XP,YP,T);END;
END;
END;
N:=POCETPRVKU(BL,BLOK);
IF N>0 THEN BEGIN
FOR M:=1 TO N DO BEGIN
NPRVEK(BL,M,BLOK);
WITH PBLOK DO BEGIN
X:=X+X1;Y:=Y+Y1;END;
BLOKKRES(PBLOK);
END;
END;

```

```

END;
END;
PROCEDURE PRLOAD;
VAR N,M,DIF:INTEGER;
NAME:ARRAY[1..8] OF CHAR;
POBJ:OBJT;
BEGIN
NAME:='';
AT(20,0);WRITE('Name
');
AT(20,5);READLN;READ(NAME);
NAME(6):='.';NAME(7):='J';NAME(8):='M';
TIN(NAME,ADDR(JMENA)+6*(POCPRIK+PBLAKT)
);
N:=PBLAKT+POCPRIK+1;
WHILE (N<=JMCEKEM) AND (JMENA(N)<>
') DO N:=N+1;
N:=N-1-POCPRIK;
NAME(7):='U';NAME(8):='K';
TIN(NAME,ADDR(UK)+16*PBLAKT);
DIF:=UK(PBLAKT+1).PRVKY[LINE]-ENDOF;
IF UK(N).PRVKY[BLOK]+5*UK(N).POCET[BLOK
]-DIF<TOPMEM THEN BEGIN
FOR M:=PBLAKT+1 TO N DO BEGIN
FOR POBJ:=LINE TO BLOK DO
UK(M).PRVKY[POBJ]:=UK(M).PRVKY[POBJ]-
DIF;
END;
DIF:=UK(N).PRVKY[BLOK]+5*UK(N).POCET[BLO
K];
IF N<PBLMAX THEN BEGIN
FOR M:=N+1 TO PBLMAX DO BEGIN
FOR POBJ:=LINE TO BLOK DO UK(M).PRVKY
[POBJ]:=DIF;
END;
END;
NAME(7):='V';NAME(8):='S';
TIN(NAME,ENDOF);
PBLAKT:=N;ENDOF:=DIF;
END
ELSE BEGIN
AT(20,0);WRITE('No memory for file
');
END;
END;
PROCEDURE INIT;
VAR N:INTEGER;POBJ:OBJT;

```

```

BEGIN
FOR N:=1 TO PBLMAX DO BEGIN
FOR POBJ:=LINE TO BLOK DO BEGIN
UK(N).PRVKY[POBJ]:=OFFSET;
UK(N).POCET[POBJ]:=0;
END;
END;
ENDOF:=OFFSET;
FOR N:=1 TO JMCEKEM DO JMENA(N):='
';
OUT(127,CHR(153));
PBLAKT:=0;
END;
PROCEDURE VYPISBLOKU;
VAR N:INTEGER;
BEGIN
PAGE;
FOR N:=POCPRIK+1 TO JMCEKEM DO BEGIN
WRITE(CHR(20),CHR(1));
WRITE((N-POCPRIK):2,CHR(20),CHR(0));WR
ITE(JMENA(N));END;
END;
PROCEDURE PROVAZENI;
VAR VX,VY,SCALE:REAL;
MFY,MFY,RYCHLOST,OTC:INTEGER;
POCETKB:INTEGER;
CKB:ARRAY[1..10] OF INTEGER;
N:INTEGER;
BEGIN
REPEAT
INIT;
PAGE;
REPEAT
WRITELN('Loading file for drawing');WR
ITELN;WRITELN;WRITELN;
PRLOAD;
WRITELN('All files loaded? (Y/N) ');
REPEAT UNTIL INCH<>CHR(0);
UNTIL INCH='Y';
OUT(127,CHR(153));
WRITELN('Parameters of drawing');WRITELN
;
WRITE('Original point X.. ');READLN;REA
D(VX);
WRITE('... Y.. ');READLN;REA
D(VY);
WRITELN;WRITE('Scale ..... ');RE
ADLN;READ(SCALE);

```

```

WRITELN;WRITE('Orientation ..... ');RE
ADLN;READ(OTC);
WRITELN;WRITE('Speed [X] ..... ');RE
ADLN;
READ(RYCHLOST);WRITELN;WRITELN('Set ori
ginal point (0..exit)');
POKE(*EA70,CHR(223));POKE(*EA7C,CHR(215
));POKE(*EA80,CHR(207));
POKE(*EA84,CHR(199));
IF OTC=3 THEN BEGIN POKE(*EA70,CHR(215
));POKE(*EA7C,CHR(223));END;
IF OTC=2 THEN BEGIN POKE(*EA70,CHR(207
));POKE(*EA7C,CHR(199));
POKE(*EA80,CHR(215));POKE(*EA
84,CHR(223));END;
USER (*EC54);
MF;
MFY:=PEEK(*EC66,INTEGER);
MFY:=PEEK((MFY+2),INTEGER);MFY:=PEEK(MF
X,INTEGER);
OG(MFY-SCALE*VX,MFY-SCALE*VY);
SC(SCALE);SPEED(ROUND(12300/RYCHLOST));
VYPISBLOKU;WRITELN;WRITELN;
REPEAT
WRITE('Number of blocks .. ');READLN;REA
D(POCETKB);
WRITELN;WRITELN;
UNTIL (POCETKB>0) AND (POCETKB<11);
WRITELN;WRITELN;
FOR N:=1 TO POCETKB DO BEGIN
AT(20,0);WRITE('
');
AT(20,0);WRITE(N,'... ');READLN;READ(
CKB(N));
END;
WRITELN;WRITELN('I am drawing');WRITELN
;
FOR N:=1 TO POCETKB DO BEGIN
PBLOK.X:=0;PBLOK.Y:=0;PBLOK.CBL:=CHR(C
KB(N));
BLOKKRES(PBLOK);
END;
WRITELN('Load a new file (Y/N)');
REPEAT UNTIL INCH<>CHR(0);
UNTIL INCH='N';
END;
BEGIN
PROVAZENI;
END.

```

## BT 100 – ZX SPECTRUM

Obslužný program pro spolupráci tiskárny BT 100  
s počítačem ZX Spectrum

Ing. Jiří Kohout, Na Švihance 12, 120 00 Praha 2

V příručce dodávané s tiskárnou BT 100 je uveden i program pro její obsluhu. Délka tohoto programu je přibližně 1kB, dalších 500 bajtů zabírá vyrovnávací paměť tiskárny. Umožňuje použití příkazů LLIST a LPRINT a kopii obrazovky. Program však neumí tisknout implicitně definované grafické znaky a znaky definované uživatelem v oblasti UDG a v příkazu LPRINT nemá příkaz TAB pro tabelaci výpisu účinek. Proto byl vytvořen program, který uvedené nedostatky odstraňuje a navíc je podstatně kratší.

Program pro spolupráci tiskárny BT 100 s mikropočítačem ZX Spectrum je dlouhý 706 bajtů, používá standardních příkazů LLIST a LPRINT (LPRINT i ve spojení s příkazem TAB), tiskne všechny grafické znaky, písmo lze zvolit s normálním sklonem nebo šikmým. Typ písma může být změněn uložením vlastního souboru znaků do RAM a příslušnou změnou systémové proměnné CHARS (zde je uložena adresa počátku tabulky znaků minus 256). Ukázka různých typů písma je na obr. 1. Program též umožňuje kopii obrazovky, příklad je na obr. 2.

Jako vyrovnávací paměť pro tiskárnu je využita část obrazové paměti (horní třetina obrazovky), která při obsluze tiskárny většinou nebyvá využita; navíc je na obrazovce vidět, co se právě tiskne.

Tiskárna je připojena k počítači přes rozhraní s obvodem 8255 (zapojení bylo v AR již několikrát publikováno, lze též použít výrobek Tesly Kolín, který je prodáván pod označením UR-4). Pro komunikaci počítač-tiskárna je použit kanál C, kde bity C0 až C3 jsou naprogramovány jako výstupní a bity C4 až C7 jako vstupní.

Nyní stručně k vlastnímu programu (viz Vypis 1). Řádky 140 až 250 (rutina INIT) naprogramují obvod 8255, do tabulky kanálových informací na adresu 23749 (#5CC5) uloží startovací adresu obslužného programu (při prvním vstupu adresu START1, kde se vymaže horní třetina obrazovky, při dalších vstupech pak adresu START) a nastaví počítač do řádků.

Řádky 450 až 1060 rozliší podle kódu v registru A o jaký typ znaku se jedná a do registrového páru DE uloží adresu prvního bajtu znaku.

Kódy menší než 32 jsou ignorovány s výjimkou hodnoty 13 – konec řádku, 23 – TAB a 20 – INVERSE. Kódy 32 až 127 jsou standardní znaky ASCII, adresa prvního bajtu znaku se vytvoří na řádcích 1030 až 1060. Kódy 128 až 143 reprezentují implicitně definované grafické znaky, znak se tvoří pomocí rutiny v ROM začínající na adrese #0B38. Znak je pak uložen v systémové proměnné MEMBOT, kde začíná zásobníková paměť kalkulátoru. Tato rutina je na řádcích 730 až 750. Kódy 144 až 164 určují znaky definované uživatelem v oblasti RAM, tuto oblast adresuje systémová proměnná UDG. Adresa prvního bajtu příslušného znaku je určena na řádcích 680 až 710. 165 až 255 jsou kódy pro tokens, tvorbu jednotlivých znaků zajišťuje rutina ROM na adrese #09F4 (řádky 600 až 610).

Rutina ULOZ na řádcích 1070 až 1280 uloží jednotlivé bajty znaků do obrazové paměti, která slouží jako vyrovnávací paměť pro tiskárnu.

Na řádcích 1190 až 1280 se kontroluje, zda není překročen povolený počet znaků na řádek.

Na řádku 1290 se rozhoduje, zda se znaky budou tisknout normálně nebo šikmo. Při tisku šikmých znaků se před vstupem do vlastní tiskové rutiny provede posun jednotlivých bitů ve vyrovnávací paměti o 1 až 7 tiskových bodů vpravo – řádky 1520 až 1760.

Na řádcích 1320 až 1500 je volána tisková rutina TISK, po vytisknutí celého řádku je vynulována vyrovnávací paměť a je testován počet vytisknutých řádků. Při překročení předem nastaveného počtu se program zastaví a po stisknutí libovolné klávesy pokračuje v tisku – řádky 1400 až 1500. Vytisknutí celé stránky je signalizováno pípním.

Rutina na řádcích 1780 až 1840 vypočítá adresu prvního bajtu znaku. Při vstupu do rutiny je v registrovém páru BC adresa začátku příslušné tabulky znaků (v ROM nebo v RAM) a v registru A je pořadové číslo znaku (první znak v tabulce má pořadové číslo nula).

Na řádcích 1870 až 1940 je vynulována vyrovnávací paměť.

Vlastní tisková rutina TISK začíná na řádku 1960. Nejprve se provede návrat vozíku a pak se testují vždy dva nad sebou ležící mikrořádky vyrovnávací paměti (jeden tiskový řádek se skládá z osmi mikrořádků). Pokud jsou oba nulové, posune se dvakrát papír v tiskárně a pokračuje se testem dalších dvou mikrořádků. Je-li alespoň jeden bajt v některém mikrořádku různý od nuly

Obr. 2. Ukázka kopie obrazovky (902-2)



(tzn. že je třeba tisknout), je v registrech D a E na řádku 2140 počet znaků zleva, které je třeba vytisknout. Vlastní test se provádí rutinou TEST na řádcích 3070 až 3200.

Řádky 2150 až 2630 vytisknou celý tiskový řádek (osm mikrořádků). Tiskne se střídavě lichý mikrořádek (při posunu vozíku zleva doprava) a sudý mikrořádek (při posunu vozíku zprava doleva).

Stiskne-li se tlačítko Q, tisk se zastaví po vytisknutí celého tiskového řádku a provede se skok do BASIC (řádky 2640 až 2690). Změnou hodnoty na řádku 2650 (v instrukci CP 81) lze změnit klávesu, kterou se zastavuje tisk.

Tisk jednoho bajtu provádí rutina na řádcích 2740 až 2930. Bajt je rotován do indikátoru CY buď vpravo nebo vlevo v závislosti na směru tisku (řádek 2750).

Rutina GR na řádcích 2950 až 3050 provede návrat vozíku, rutina PS.PAP (řádky 3220 až 3320) posune papír v tiskárně o jeden mikrořádek.

Rutiny C60 (řádky 3340 až 3400) a C70 (řádky 3420 až 3600) synchronizují tisk. O poloze vozíku tiskárny informují bity 7 a 6 na portu 95. Je-li bit 7 roven nule, vytiskne se jeden bod. Bit 6 slouží k určení začátku mikrořádku při tisku zprava i zleva.

Rutina na řádcích 3620 až 3660 jemně nastavuje synchronizaci tisku svislých čar.

Změnou hodnoty v registru A. popřípadě přidáním dalších instrukcí NOP lze nastavit přesnou polohu vytisknutých bodů. Záleží na rychlosti posuvu vozíku, uvedené hodnoty

vyhovovaly při rychlosti tisku 170 bodů/s. Rychlost vozíku lze nastavit trimrem na desce motorů (deska je umístěna podél přední strany tiskárny).

Obrazovku kopíruje program COPY na řádcích 280 až 400. Nejprve se provede inicializace tiskárny a nastaví se šířka tisku na 32 znaků/řádek. Potom se postupně vytiskne všech 24 tiskových řádků obrazovky (vždy po vytisknutí jednoho tiskového řádku rutina POSUV na řádcích 3680 až 4090 posune do bafu další tiskový řádek), konec tisku je signalizován pípním a po stisku libovolné klávesy se provede návrat do BASICu. Je-li program nahrán od adresy A (v uvedeném případě A=64000), inicializace se provede příkazem **RANDOMIZE USR A**, pak již mohou následovat BASIC příkazy **LLIST** nebo **LPRINT**. Formát tisku lze nastavit uložením příslušných hodnot na následující adresy:

počet řádků na stránku	POKE A+22, počet řádků
počet znaků na řádek	POKE A+216, počet znaků
písmo normal	POKE A+225,0
písmo šikmé	POKE A+225,46
synchronizace tisku	POKE A+626, hodnota

Obrazovka se zkopíruje příkazem **RANDOMIZE USR A+2**.

Program spolupracuje i s assemblerem GENS3, obslužný program pro tiskárnu pak lze zkrátit asi na 540 bajtů vypuštěním části pro kopírování obrazovky, tisk grafických znaků a tisk šikmého písma. Konkrétně se

1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

Obr. 1. Ukázka různých typů písma (902-1)

vypusti řádky 40, 50, 110, 120, 230-250, 280-400, 460-530, 560-590, 640-1010, 1520-1760, 3680-4090, 4130 a 4140. Po tomto zkrácení programu se však změní výše uvedené adresy pro nastavení formátu tisku; úpravu si provede již jistě každý sám.

### Výpis 1. Obslužný program (902-V1)

```

10 *C-
20 ;BT 100 pro LLIST,
30
40000 000 UDG EQU 23575
40000 000 MEMBOT EQU 23580
40000 000 CHARS EQU 23585
40000 000 LAST_K EQU 23590
40000 000 BAFA EQU 40000
500
FA000 1000 ORG 64000
FA000 1100 JR INIT
FA002 1200 JR COPY
1300
FA004 1400 INIT CALL UL_BFR
FA007 1500 LD A,154
FA009 1600 LD A,#7F,A
FA00B 1700 LD A,255
FA00D 1800 LD A,(95),A
FA00F 1900 LD HL,START1
FA012 2000 LD A,#5CC5,HL
FA015 2100 INIRAD LD
;Pocet radku
FA017 2200 NUL_AD LD (P_RAD),A
FA01H 2300 XOR A
FA01B 2400 LD (AD_TAB),A
FA01E 2500 LD (AD_VID),A
FA021 2600 RET
2700
FA022 2800 COPY CALL INIT
FA025 2900 LD A,32
FA027 3000 LD (A_COPY),A
FA029 3100 LD B,24
FA02C 3200 TCOPY PUSH BC
FA02D 3300 CALL TISK
FA030 3400 CALL POSUV
FA033 3500 POP BC
FA034 3600 OJNZ TCOPY
FA036 3700 LD A,59
FA038 3800 LD (A_COPY),A
FA03B 3900 CALL KEY
FA03E 4000 RET
4100
FA03F 4200 START1 CALL NULBFR
FA042 4300 LD HL,START
FA045 4400 LD HL,(23749),HL
FA048 4500 START EH
FA049 4600 LD E,A
FA04B 4700 LD A,(AD_TAB)
FA04D 4800 LD A,(AD_VID)
FA04F 4900 LD A,(AD_VID)
FA051 5000 LD A,(AD_VID)
FA054 5100 LD A,(AD_VID)
FA056 5200 LD A,(AD_VID)
FA058 5300 LD A,(AD_VID)
FA05B 5400 LD A,(AD_VID)
FA05E 5500 LD A,(AD_VID)
FA060 5600 LD A,(AD_VID)
FA062 5700 LD A,(AD_VID)
FA064 5800 LD A,(AD_VID)
FA066 5900 LD A,(AD_VID)
FA068 6000 LD A,(AD_VID)
FA06A 6100 LD A,(AD_VID)
FA06C 6200 LD A,(AD_VID)
FA06E 6300 LD A,(AD_VID)
FA070 6400 LD A,(AD_VID)
FA072 6500 LD A,(AD_VID)
FA074 6600 LD A,(AD_VID)
FA076 6700 LD A,(AD_VID)
FA078 6800 LD A,(AD_VID)
FA07A 6900 LD A,(AD_VID)
FA07C 7000 LD A,(AD_VID)
FA07E 7100 LD A,(AD_VID)
7200
FA081 7300 IMP_GR LD B,A
FA083 7400 CALL #0B38
FA085 7500 LD DE,MEMBOT
FA088 7600 LD ULOZ
7700
FA089 7800 TAB LD A,1
FA092 7900 LD (AD_TAB),A
FA094 8000 RET
8100
FA095 8200 TAB1 LD D,0
FA098 8300 LD E,A
FA09B 8400 LD A,E
FA09D 8500 LD A,E
FA09F 8600 LD A,E
FA0A1 8700 LD A,E
FA0A3 8800 LD A,E
FA0A5 8900 LD A,E
FA0A7 9000 LD A,E
FA0A9 9100 LD A,E
FA0AB 9200 LD A,E
FA0AD 9300 LD A,E
FA0AF 9400 LD A,E
FA0B1 9500 LD A,E
FA0B3 9600 LD A,E
FA0B5 9700 LD A,E
FA0B7 9800 LD A,E
FA0B9 9900 LD A,E
FA0BB 1000 LD A,E
FA0BD 1010 LD A,E
FA0BF 1020 LD A,E
FA0C1 1030 LD A,E
FA0C3 1040 LD A,E
FA0C5 1050 LD A,E
FA0C7 1060 LD A,E
FA0C9 1070 LD A,E
FA0CB 1080 LD A,E
FA0CD 1090 LD A,E
FA0CF 1100 LD A,E
FA0D1 1110 LD A,E
FA0D3 1120 LD A,E
FA0D5 1130 LD A,E
FA0D7 1140 LD A,E
FA0D9 1150 LD A,E
FA0DB 1160 LD A,E
FA0DD 1170 LD A,E
FA0DF 1180 LD A,E
FA0E1 1190 LD A,E
FA0E3 1200 LD A,E
FA0E5 1210 LD A,E
FA0E7 1220 LD A,E
FA0E9 1230 LD A,E
FA0EB 1240 LD A,E
FA0ED 1250 LD A,E
FA0EF 1260 LD A,E
FA0F1 1270 LD A,E
FA0F3 1280 LD A,E
FA0F5 1290 LD A,E
FA0F7 1300 LD A,E
FA0F9 1310 LD A,E
FA0FB 1320 LD A,E
FA0FD 1330 LD A,E
FA0FF 1340 LD A,E
FA101 1350 LD A,E
FA103 1360 LD A,E
FA105 1370 LD A,E
FA107 1380 LD A,E
FA109 1390 LD A,E
FA10B 1400 LD A,E
FA10D 1410 LD A,E
FA10F 1420 LD A,E
FA111 1430 LD A,E
FA113 1440 LD A,E
FA115 1450 LD A,E
FA117 1460 LD A,E
FA119 1470 LD A,E
FA11B 1480 LD A,E
FA11D 1490 LD A,E
FA11F 1500 LD A,E
FA121 1510 LD A,E
FA123 1520 LD A,E
FA125 1530 LD A,E
FA127 1540 LD A,E
FA129 1550 LD A,E
FA12B 1560 LD A,E
FA12D 1570 LD A,E
FA12F 1580 LD A,E
FA131 1590 LD A,E
FA133 1600 LD A,E
FA135 1610 LD A,E
FA137 1620 LD A,E
FA139 1630 LD A,E
FA13B 1640 LD A,E
FA13D 1650 LD A,E
FA13F 1660 LD A,E
FA141 1670 LD A,E
FA143 1680 LD A,E
FA145 1690 LD A,E
FA147 1700 LD A,E
FA149 1710 LD A,E
FA14B 1720 LD A,E
FA14D 1730 LD A,E
FA14F 1740 LD A,E
FA151 1750 LD A,E
FA153 1760 LD A,E
FA155 1770 LD A,E
FA157 1780 LD A,E
FA159 1790 LD A,E
FA15B 1800 LD A,E
FA15D 1810 LD A,E
FA15F 1820 LD A,E
FA161 1830 LD A,E
FA163 1840 LD A,E
FA165 1850 LD A,E
FA167 1860 LD A,E
FA169 1870 LD A,E
FA16B 1880 LD A,E
FA16D 1890 LD A,E
FA16F 1900 LD A,E
FA171 1910 LD A,E
FA173 1920 LD A,E
FA175 1930 LD A,E
FA177 1940 LD A,E
FA179 1950 LD A,E
FA17B 1960 LD A,E
FA17D 1970 LD A,E
FA17F 1980 LD A,E
FA181 1990 LD A,E
FA183 2000 LD A,E
FA185 2010 LD A,E
FA187 2020 LD A,E
FA189 2030 LD A,E
FA18B 2040 LD A,E
FA18D 2050 LD A,E
FA18F 2060 LD A,E
FA191 2070 LD A,E
FA193 2080 LD A,E
FA195 2090 LD A,E
FA197 2100 LD A,E
FA199 2110 LD A,E
FA19B 2120 LD A,E
FA19D 2130 LD A,E
FA19F 2140 LD A,E
FA1A1 2150 LD A,E
FA1A3 2160 LD A,E
FA1A5 2170 LD A,E
FA1A7 2180 LD A,E
FA1A9 2190 LD A,E
FA1AB 2200 LD A,E
FA1AD 2210 LD A,E
FA1AF 2220 LD A,E
FA1B1 2230 LD A,E
FA1B3 2240 LD A,E
FA1B5 2250 LD A,E
FA1B7 2260 LD A,E
FA1B9 2270 LD A,E
FA1BB 2280 LD A,E
FA1BD 2290 LD A,E
FA1BF 2300 LD A,E
FA1C1 2310 LD A,E
FA1C3 2320 LD A,E
FA1C5 2330 LD A,E
FA1C7 2340 LD A,E
FA1C9 2350 LD A,E
FA1CB 2360 LD A,E
FA1CD 2370 LD A,E
FA1CF 2380 LD A,E
FA1D1 2390 LD A,E
FA1D3 2400 LD A,E
FA1D5 2410 LD A,E
FA1D7 2420 LD A,E
FA1D9 2430 LD A,E
FA1DB 2440 LD A,E
FA1DD 2450 LD A,E
FA1DF 2460 LD A,E
FA1E1 2470 LD A,E
FA1E3 2480 LD A,E
FA1E5 2490 LD A,E
FA1E7 2500 LD A,E
FA1E9 2510 LD A,E
FA1EB 2520 LD A,E
FA1ED 2530 LD A,E
FA1EF 2540 LD A,E
FA1F1 2550 LD A,E
FA1F3 2560 LD A,E
FA1F5 2570 LD A,E
FA1F7 2580 LD A,E
FA1F9 2590 LD A,E
FA1FB 2600 LD A,E
FA1FD 2610 LD A,E
FA1FF 2620 LD A,E
FA201 2630 LD A,E
FA203 2640 LD A,E
FA205 2650 LD A,E
FA207 2660 LD A,E
FA209 2670 LD A,E
FA20B 2680 LD A,E
FA20D 2690 LD A,E
FA20F 2700 LD A,E
FA211 2710 LD A,E
FA213 2720 LD A,E
FA215 2730 LD A,E
FA217 2740 LD A,E
FA219 2750 LD A,E
FA21B 2760 LD A,E
FA21D 2770 LD A,E
FA21F 2780 LD A,E
FA221 2790 LD A,E
FA223 2800 LD A,E
FA225 2810 LD A,E
FA227 2820 LD A,E
FA229 2830 LD A,E
FA22B 2840 LD A,E
FA22D 2850 LD A,E
FA22F 2860 LD A,E
FA231 2870 LD A,E
FA233 2880 LD A,E
FA235 2890 LD A,E
FA237 2900 LD A,E
FA239 2910 LD A,E
FA23B 2920 LD A,E
FA23D 2930 LD A,E
FA23F 2940 LD A,E
FA241 2950 LD A,E
FA243 2960 LD A,E
FA245 2970 LD A,E
FA247 2980 LD A,E
FA249 2990 LD A,E
FA24B 3000 LD A,E
FA24D 3010 LD A,E
FA24F 3020 LD A,E
FA251 3030 LD A,E
FA253 3040 LD A,E
FA255 3050 LD A,E
FA257 3060 LD A,E
FA259 3070 LD A,E
FA25B 3080 LD A,E
FA25D 3090 LD A,E
FA25F 3100 LD A,E
FA261 3110 LD A,E
FA263 3120 LD A,E
FA265 3130 LD A,E
FA267 3140 LD A,E
FA269 3150 LD A,E
FA26B 3160 LD A,E
FA26D 3170 LD A,E
FA26F 3180 LD A,E
FA271 3190 LD A,E
FA273 3200 LD A,E
FA275 3210 LD A,E
FA277 3220 LD A,E
FA279 3230 LD A,E
FA27B 3240 LD A,E
FA27D 3250 LD A,E
FA27F 3260 LD A,E
FA281 3270 LD A,E
FA283 3280 LD A,E
FA285 3290 LD A,E
FA287 3300 LD A,E
FA289 3310 LD A,E
FA28B 3320 LD A,E
FA28D 3330 LD A,E
FA28F 3340 LD A,E
FA291 3350 LD A,E
FA293 3360 LD A,E
FA295 3370 LD A,E
FA297 3380 LD A,E
FA299 3390 LD A,E
FA29B 3400 LD A,E
FA29D 3410 LD A,E
FA29F 3420 LD A,E
FA2A1 3430 LD A,E
FA2A3 3440 LD A,E
FA2A5 3450 LD A,E
FA2A7 3460 LD A,E
FA2A9 3470 LD A,E
FA2AB 3480 LD A,E
FA2AD 3490 LD A,E
FA2AF 3500 LD A,E
FA2B1 3510 LD A,E
FA2B3 3520 LD A,E
FA2B5 3530 LD A,E
FA2B7 3540 LD A,E
FA2B9 3550 LD A,E
FA2BB 3560 LD A,E
FA2BD 3570 LD A,E
FA2BF 3580 LD A,E
FA2C1 3590 LD A,E
FA2C3 3600 LD A,E
FA2C5 3610 LD A,E
FA2C7 3620 LD A,E
FA2C9 3630 LD A,E
FA2CB 3640 LD A,E
FA2CD 3650 LD A,E
FA2CF 3660 LD A,E
FA2D1 3670 LD A,E
FA2D3 3680 LD A,E
FA2D5 3690 LD A,E
FA2D7 3700 LD A,E
FA2D9 3710 LD A,E
FA2DB 3720 LD A,E
FA2DD 3730 LD A,E
FA2DF 3740 LD A,E
FA2E1 3750 LD A,E
FA2E3 3760 LD A,E
FA2E5 3770 LD A,E
FA2E7 3780 LD A,E
FA2E9 3790 LD A,E
FA2EB 3800 LD A,E
FA2ED 3810 LD A,E
FA2EF 3820 LD A,E
FA2F1 3830 LD A,E
FA2F3 3840 LD A,E
FA2F5 3850 LD A,E
FA2F7 3860 LD A,E
FA2F9 3870 LD A,E
FA2FB 3880 LD A,E
FA2FD 3890 LD A,E
FA2FF 3900 LD A,E
FA301 3910 LD A,E
FA303 3920 LD A,E
FA305 3930 LD A,E
FA307 3940 LD A,E
FA309 3950 LD A,E
FA30B 3960 LD A,E
FA30D 3970 LD A,E
FA30F 3980 LD A,E
FA311 3990 LD A,E
FA313 4000 LD A,E
FA315 4010 LD A,E
FA317 4020 LD A,E
FA319 4030 LD A,E
FA31B 4040 LD A,E
FA31D 4050 LD A,E
FA31F 4060 LD A,E
FA321 4070 LD A,E
FA323 4080 LD A,E
FA325 4090 LD A,E
FA327 4100 LD A,E
FA329 4110 LD A,E
FA32B 4120 LD A,E
FA32D 4130 LD A,E
FA32F 4140 LD A,E
FA331 4150 LD A,E
FA333 4160 LD A,E
FA335 4170 LD A,E
FA337 4180 LD A,E
FA339 4190 LD A,E
FA33B 4200 LD A,E
FA33D 4210 LD A,E
FA33F 4220 LD A,E
FA341 4230 LD A,E
FA343 4240 LD A,E
FA345 4250 LD A,E
FA347 4260 LD A,E
FA349 4270 LD A,E
FA34B 4280 LD A,E
FA34D 4290 LD A,E
FA34F 4300 LD A,E
FA351 4310 LD A,E
FA353 4320 LD A,E
FA355 4330 LD A,E
FA357 4340 LD A,E
FA359 4350 LD A,E
FA35B 4360 LD A,E
FA35D 4370 LD A,E
FA35F 4380 LD A,E
FA361 4390 LD A,E
FA363 4400 LD A,E
FA365 4410 LD A,E
FA367 4420 LD A,E
FA369 4430 LD A,E
FA36B 4440 LD A,E
FA36D 4450 LD A,E
FA36F 4460 LD A,E
FA371 4470 LD A,E
FA373 4480 LD A,E
FA375 4490 LD A,E
FA377 4500 LD A,E
FA379 4510 LD A,E
FA37B 4520 LD A,E
FA37D 4530 LD A,E
FA37F 4540 LD A,E
FA381 4550 LD A,E
FA383 4560 LD A,E
FA385 4570 LD A,E
FA387 4580 LD A,E
FA389 4590 LD A,E
FA38B 4600 LD A,E
FA38D 4610 LD A,E
FA38F 4620 LD A,E
FA391 4630 LD A,E
FA393 4640 LD A,E
FA395 4650 LD A,E
FA397 4660 LD A,E
FA399 4670 LD A,E
FA39B 4680 LD A,E
FA39D 4690 LD A,E
FA39F 4700 LD A,E
FA3A1 4710 LD A,E
FA3A3 4720 LD A,E
FA3A5 4730 LD A,E
FA3A7 4740 LD A,E
FA3A9 4750 LD A,E
FA3AB 4760 LD A,E
FA3AD 4770 LD A,E
FA3AF 4780 LD A,E
FA3B1 4790 LD A,E
FA3B3 4800 LD A,E
FA3B5 4810 LD A,E
FA3B7 4820 LD A,E
FA3B9 4830 LD A,E
FA3BB 4840 LD A,E
FA3BD 4850 LD A,E
FA3BF 4860 LD A,E
FA3C1 4870 LD A,E
FA3C3 4880 LD A,E
FA3C5 4890 LD A,E
FA3C7 4900 LD A,E
FA3C9 4910 LD A,E
FA3CB 4920 LD A,E
FA3CD 4930 LD A,E
FA3CF 4940 LD A,E
FA3D1 4950 LD A,E
FA3D3 4960 LD A,E
FA3D5 4970 LD A,E
FA3D7 4980 LD A,E
FA3D9 4990 LD A,E
FA3DB 5000 LD A,E
FA3DD 5010 LD A,E
FA3DF 5020 LD A,E
FA3E1 5030 LD A,E
FA3E3 5040 LD A,E
FA3E5 5050 LD A,E
FA3E7 5060 LD A,E
FA3E9 5070 LD A,E
FA3EB 5080 LD A,E
FA3ED 5090 LD A,E
FA3EF 5100 LD A,E
FA3F1 5110 LD A,E
FA3F3 5120 LD A,E
FA3F5 5130 LD A,E
FA3F7 5140 LD A,E
FA3F9 5150 LD A,E
FA3FB 5160 LD A,E
FA3FD 5170 LD A,E
FA3FF 5180 LD A,E
FA401 5190 LD A,E
FA403 5200 LD A,E
FA405 5210 LD A,E
FA407 5220 LD A,E
FA409 5230 LD A,E
FA40B 5240 LD A,E
FA40D 5250 LD A,E
FA40F 5260 LD A,E
FA411 5270 LD A,E
FA413 5280 LD A,E
FA415 5290 LD A,E
FA417 5300 LD A,E
FA419 5310 LD A,E
FA41B 5320 LD A,E
FA41D 5330 LD A,E
FA41F 5340 LD A,E
FA421 5350 LD A,E
FA423 5360 LD A,E
FA425 5370 LD A,E
FA427 5380 LD A,E
FA429 5390 LD A,E
FA42B 5400 LD A,E
FA42D 5410 LD A,E
FA42F 5420 LD A,E
FA431 5430 LD A,E
FA433 5440 LD A,E
FA435 5450 LD A,E
FA437 5460 LD A,E
FA439 5470 LD A,E
FA43B 5480 LD A,E
FA43D 5490 LD A,E
FA43F 5500 LD A,E
FA441 5510 LD A,E
FA443 5520 LD A,E
FA445 5530 LD A,E
FA447 5540 LD A,E
FA449 5550 LD A,E
FA44B 5560 LD A,E
FA44D 5570 LD A,E
FA44F 5580 LD A,E
FA451 5590 LD A,E
FA453 5600 LD A,E
FA455 5610 LD A,E
FA457 5620 LD A,E
FA459 5630 LD A,E
FA45B 5640 LD A,E
FA45D 5650 LD A,E
FA45F 5660 LD A,E
FA461 5670 LD A,E
FA463 5680 LD A,E
FA465 5690 LD A,E
FA467 5700 LD A,E
FA469 5710 LD A,E
FA46B 5720 LD A,E
FA46D 5730 LD A,E
FA46F 5740 LD A,E
FA471 5750 LD A,E
FA473 5760 LD A,E
FA475 5770 LD A,E
FA477 5780 LD A,E
FA479 5790 LD A,E
FA47B 5800 LD A,E
FA47D 5810 LD A,E
FA47F 5820 LD A,E
FA481 5830 LD A,E
FA483 5840 LD A,E
FA485 5850 LD A,E
FA487 5860 LD A,E
FA489 5870 LD A,E
FA48B 5880 LD A,E
FA48D 5890 LD A,E
FA48F 5900 LD A,E
FA491 5910 LD A,E
FA493 5920 LD A,E
FA495 5930 LD A,E
FA497 5940 LD A,E
FA499 5950 LD A,E
FA49B 5960 LD A,E
FA49D 5970 LD A,E
FA49F 5980 LD A,E
FA4A1 5990 LD A,E
FA4A3 6000 LD A,E
FA4A5 6010 LD A,E
FA4A7 6020 LD A,E
FA4A9 6030 LD A,E
FA4AB 6040 LD A,E
FA4AD 6050 LD A,E
FA4AF 6060 LD A,E
FA4B1 6070 LD A,E
FA4B3 6080 LD A,E
FA4B5 6090 LD A,E
FA4B7 6100 LD A,E
FA4B9 6110 LD A,E
FA4BB 6120 LD A,E
FA4BD 6130 LD A,E
FA4BF 6140 LD A,E
FA4C1 6150 LD A,E
FA4C3 6160 LD A,E
FA4C5 6170 LD A,E
FA4C7 6180 LD A,E
FA4C9 6190 LD A,E
FA4CB 6200 LD A,E
FA4CD 6210 LD A,E
FA4CF 6220 LD A,E
FA4D1 6230 LD A,E
FA4D3 6240 LD A,E
FA4D5 6250 LD A,E
FA4D7 6260 LD A,E
FA4D9 6270 LD A,E
FA4DB 6280 LD A,E
FA4DD 6290 LD A,E
FA4DF 6300 LD A,E
FA4E1 6310 LD A,E
FA4E3 6320 LD A,E
FA4E5 6330 LD A,E
FA4E7 6340 LD A,E
FA4E9 6350 LD A,E
FA4EB 6360 LD A,E
FA4ED 6370 LD A,E
FA4EF 6380 LD A,E
FA4F1 6390 LD A,E
FA4F3 6400 LD A,E
FA4F5 6410 LD A,E
FA4F7 6420 LD A,E
FA4F9 6430 LD A,E
FA4FB 6440 LD A,E
FA4FD 6450 LD A,E
FA4FF 6460 LD A,E
FA501 6470 LD A,E
FA503 6480 LD A,E
FA505 6490 LD A,E
FA507 6500 LD A,E
FA509 6510 LD A,E
FA50B 6520 LD A,E
FA50D 6530 LD A,E
FA50F 6540 LD A,E
FA511 6550 LD A,E
FA513 6560 LD A,E
FA515 6570 LD A,E
FA517 6580 LD A,E
FA519 6590 LD A,E
FA51B 6600 LD A,E
FA51D 6610 LD A,E
FA51F 6620 LD A,E
FA521 6630 LD A,E
FA523 6640 LD A,E
FA525 6650 LD A,E
FA527 6660 LD A,E
FA529 6670 LD A,E
FA52B 6680 LD A,E
FA52D 6690 LD A,E
FA52F 6700 LD A,E
FA531 6710 LD A,E
FA533 6720 LD A,E
FA535 6730 LD A,E
FA537 6740 LD A,E
FA539 6750 LD A,E
FA53B 6760 LD A,E
FA53D 6770 LD A,E
FA53F 6780 LD A,E
FA541 6790 LD A,E
FA543 6800 LD A,E
FA545 6810 LD A,E
FA547 6820 LD A,E
FA549 6830 LD A,E
FA54B 6840 LD A,E
FA54D 6850 LD A,E
FA54F 6860 LD A,E
FA551 6870 LD A,E
FA553 6880 LD A,E
FA555 6890 LD A,E
FA557 6900 LD A,E
FA559 6910 LD A,E
FA55B 6920 LD A,E
FA55D 6930 LD A,E
FA55F 6940 LD A,E
FA561 6950 LD A,E
FA563 6960 LD A,E
FA565 6970 LD A,E
FA567 6980 LD A,E
FA569 6990 LD A,E
FA56B 7000 LD A,E
FA56D 7010 LD A,E
FA56F 7020 LD A,E
FA571 7030 LD A,E
FA573 7040 LD A,E
FA575 7050 LD A,E
FA577 7060 LD A,E
FA579 7070 LD A,E
FA57B 7080 LD A,E
FA57D 7090 LD A,E
FA57F 7100 LD A,E
FA581 7110 LD A,E
FA583 7120 LD A,E
FA585 7130 LD A,E
FA587 7140 LD A,E
FA589 7150 LD A,E
FA58B 7160 LD A,E
FA58D 7170 LD A,E
FA58F 7180 LD A,E
FA591 7190 LD A,E
FA593 7200 LD A,E
FA595 7210 LD A,E
FA597 7220 LD A,E
FA599 7230 LD A,E
FA59B 7240 LD A,E
FA59D 7250 LD A,E
FA59F 7260 LD A,E
FA5A1 7270 LD A,E
FA5A3 7280 LD A,E
FA5A5 7290 LD A,E
FA5A7 7300 LD A,E
FA5A9 7310 LD A,E
FA5AB 7320 LD A,E
FA5AD 7330 LD A,E
FA5AF 7340 LD A,E
FA5B1 7350 LD A,E
FA5B3 7360 LD A,E
FA5B5 7370 LD A,E
FA5B7 7380 LD A,E
FA5B9 7390 LD A,E
FA5BB 7400 LD A,E
FA5BD 7410 LD A,E
FA5BF 7420 LD A,E
FA5C1 7430 LD A,E
FA5C3 7440 LD A,E
FA5C5 7450 LD A,E
FA5C7 7460 LD A,E
FA5C9 7470 LD A,E
FA5CB 7480 LD A,E
FA5CD 7490 LD A,E
FA5CF 7500 LD A,E
FA5D1 7510 LD A,E
FA5D3 7520 LD A,E
FA5D5 7530 LD A,E
FA5D7 7540 LD A,E
FA5D9 7550 LD A,E
FA5DB 7560 LD A,E
FA5DD 7570 LD A,E
FA5DF 7580 LD A,E
FA5E1 7590 LD A,E
FA5E3 7600 LD A,E
FA5E5 7610 LD A,E
FA5E7 7620 LD A,E
FA5E9 7630 LD A,E
FA5EB 7640 LD A,E
FA5ED 7650 LD A,E
FA5EF 7660 LD A,E
FA5F1 7670 LD A,E
FA5F3 7680 LD A,E
FA5F5 7690 LD A,E
FA5F7 7700 LD A,E
FA5F9 7710 LD A,E
FA5FB 7720 LD A,E
FA5FD 7730 LD A,E
FA5FF 7740 LD A,E
FA601 7750 LD A,E
FA603 7760 LD A,E
FA605 7770 LD A,E
FA607 7780 LD A,E
FA609 7790 LD A,E
FA60B 7800 LD A,E
FA60D 7810 LD A,E
FA60F 7820 LD A,E
FA611 7830 LD A,E
FA613 7840 LD A,E
FA615 7850 LD A,E
FA617 7860 LD A,E
FA619 7870 LD A,E
FA61B 7880 LD A,E
FA61D 7890 LD A,E
FA61F 7900 LD A,E
FA621 7910 LD A,E
FA623 7920 LD A,E
FA625 7930 LD A,E
FA627 7940 LD A,E
FA629 7950 LD A,E
FA62B 7960 LD A,E
FA62D 7970 LD A,E
FA62F 7980 LD A,E
FA631 7990 LD A,E
FA633 8000 LD A,E
FA635 8010 LD A,E
FA637 8020 LD A,E
FA639 8030 LD A,E
FA63B 8040 LD A,E
FA63D 8050 LD A,E
FA63F 8060 LD A,E
FA641 8070 LD A,E
FA643 8080 LD A,E
FA645 8090 LD A,E
FA647 8100 LD A,E
FA649 8110 LD A,E
FA64B 8120 LD A,E
FA64D 8130 LD A,E
FA64F 8140 LD A,E
FA651 8150 LD A,E
FA653 8160 LD A,E
FA655 8170 LD A,E
FA657 8180 LD A,E
FA659 8190 LD A,E
FA65B 8200 LD A,E
FA65D 8210 LD A,E
FA65F 8220 LD A,E
FA661 8230 LD A,E
FA663 8240 LD A,E
FA665 8250 LD A,E
FA667 8260 LD A,E
FA669 8270 LD A,E
FA66B 8280 LD A,E
FA66D 8290 LD A,E
FA66F 8300 LD A,E
FA671 8310 LD A,E
FA673 8320 LD A,E
FA675 8330 LD A,E
FA677 8340 LD A,E
FA679 8350 LD A,E
FA67B 8360 LD A,E
FA67D 8370 LD A,E
FA67F 8380 LD A,E
FA681 8390 LD A,E
FA683 8400 LD A,E
FA685 8410 LD A,E
FA687 8420 LD A,E
FA689 8430 LD A,E
FA68B 8440 LD A,E
FA68D 8450 LD A,E
FA68F 8460 LD A,E
FA691 8470 LD A,E
FA693 8480 LD A,E
FA695 8490 LD A,E
FA697 8500 LD A,E
FA699 8510 LD A,E
FA69B 8520 LD A,E
FA69D 8530 LD A,E
FA69F 8540 LD A,E
FA6A1 8550 LD A,E
FA6A3 8560 LD A,E
FA6A5 8570 LD A,E
FA6A7 8580 LD A,E
FA6A9 8590 LD A,E
FA6AB 8600 LD A,E
FA6AD 8610 LD A,E
FA6AF 8620 LD A,E
FA6B1 8630 LD A,E
FA6B3 8640 LD A,E
FA6B5 8650 LD A,E
FA6B7 8660 LD A,E
FA6B9 8670 LD A,E
FA6BB 8680 LD A,E
FA6BD 8690 LD A,E
FA6BF 8700 LD A,E
FA6C1 8710 LD A,E
FA6C3 8720 LD A,E
FA6C5 8730 LD A,E
FA6C7 8740 LD A,E
FA6C9 8750 LD A,E
FA6CB 8760 LD A,E
FA6CD 8770 LD A,E
FA6CF 8780 LD A,E
FA6D1 8790 LD A,E
FA6D3 8800 LD A,E
FA6D5 8810 LD A,E
FA6D7 8820 LD A,E
FA6D9 8830 LD A,E
FA6DB 8840 LD A,E
FA6DD 8850 LD A,E
FA6DF 8860 LD A,E
FA6E1 8870 LD A,E
FA6E3 8880 LD A,E
FA6E5 8890 LD A,E
FA6E7 8900 LD A,E
FA6E9 8910 LD A,E
FA6EB 8920 LD A,E
FA6ED 8930 LD A,E
FA6EF 8940 LD A,E
FA6F1 8950 LD A,E
FA6F3 8960 LD A,E
FA6F5 8970 LD A,E
FA6F7 8980 LD A,E
FA6F9 8990 LD A,E
FA6FB 9000 LD A,E
FA6FD 9010 LD A,E
FA6FF 9020 LD A,E
FA701 9030 LD A,E
FA703 9040 LD A,E
FA705 9050 LD A,E
FA707 9060 LD A,E
FA709 9070 LD A,E
FA70B 9080 LD A,E
FA70D 9090 LD A,E
FA70F 9100 LD A,E
FA711 9110 LD A,E
FA713 9120 LD A,E
FA715 9130 LD A,E
FA717 9140 LD A,E
FA719 9150 LD A,E
FA71B 9160 LD A,E
FA71D 9170 LD A,E
FA71F 9180 LD A,E
FA721 9190 LD A,E
FA723 9200 LD A,E
FA725 9210 LD A,E
FA727 9220 LD A,E
FA729 9230 LD A,E
FA72B 9240 LD A,E
FA72D 9250 LD A,E
FA72F 9260 LD A,E
FA731 9270 LD A,E
FA733 9280 LD A,E
FA735 9290 LD A,E
FA737 9300 LD A,E
FA739 9310 LD A,E
FA73B 9320 LD A
```